MASTER EN SCIENCES INFORMATIQUES

Contribution au développement d'un système expert conseil en mise en peinture

Votot, Yvan; Wilmet, Isabelle

Award date:
1987

Awarding institution:
Universite de Namur

CONTRIBUTION AU DEVELOPPEMENT

D'UN SYSTEME EXPERT CONSEIL

EN MISE EN PEINTURE

Yvan Votot et Isabelle Wilmet

Mémoire présenté en vue de l'obtention du titre de licencié et maître en informatique

Promoteur : Monsieur A. van Lamsweerde

Année académique 1986 - 1987

Remerciements

Nous remercions vivement Monsieur Axel van Lamsweerde, professeur aux Facultés Notre-Dame de la Paix à Namur et promoteur de ce mémoire, pour les précieuses indications qu'il nous a fournies tout au long de l'élaboration de ce mémoire.

Que soient remerciés la firme Trimetal et son personnel pour l'accueil chaleureux que nous y avons reçu, et pour les moyens qui ont été mis à notre disposition.

Nous tenons à exprimer toute notre reconnaissance à Monsieur Duviviér, responsable du service informatique, pour ses nombreux conseils, à Messieurs Gossens, Collard et Offerman du laboratoire chimique et aux directeurs commerciaux pour leur disponibilité et les renseignements indispensables à la mise en oeuvre de ce système expert.

Enfin, nous adressons nos remerciements à nos parents et à toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce travail.

Tables des matières

Bibliographie

Annexes

- Le code PROLOG
- Tableau des critères commerciaux pour chaque produit
- Tableau des données utiles pour l'explication des règles
- Questions destinées aux chimistes
- Questions destinées aux commerciaux

Bien que le travail de rédaction fut réparti entre les deux auteurs, la conception de la totalité du mémoire a été réalisée par les deux.

I. INTRODUCTION

L'intelligence artificielle a remporté un succès considérable dans le développement de systèmes experts depuis le milieu des années 60. Cette partie de l'intelligence artificielle s'est concentrée sur la construction de programmes à haute performance spécialisés dans des domaines professionnels. Ces travaux ont souligné l'importance de l'expertise humaine.

Le domaine des systèmes experts étudie les méthodes et les techniques pour la construction de machines imitant l'homme avec des techniques de résolution de problèmes spécialisées dans l'expertise. Une expertise est un ensemble de connaissances sur un domaine particulier, permettant de comprendre les problèmes relatifs à ce domaine et le savoir-faire intervenant dans la résolution de ces problèmes. La puissance d'un système expert dépend de la connaissance dont il dispose et non du formalisme particulier, et du schéma d'inférence qu'il utilise.

Les systèmes experts diffèrent des systèmes conventionnels de manipulation de données et également des systèmes développés dans les autres branches de l'intelligence artificielle. En premier lieu, ils exécutent des tâches difficiles à un niveau de performance d'un expert. Ils résolvent avec succès des problèmes pour lesquels ils sont adaptés, ce qui n'est pas du tout aisé puisque dans certains cas, un problème n'a pas une réponse unique. Ensuite, ils mettent l'accent sur des stratégies de résolution de problèmes afférents à un domaine spécifique. Le comportement des systèmes experts évitent une recherche aveugle au travers d'un nombre important d'hypothèses en faveur d'une élimination rapide de nombreuses possibilités. Ils emploient également leurs propres connaissances pour raisonner sur leur processus d'inférence et fournissent des explications ou des justifications aux conclusions atteintes. Enfin, ils résolvent des problèmes qui généralement tombent dans une des catégories décrites ci-dessous.

- Les systèmes d'interprétation : ils expliquent des données observées en leur assignant des significations symboliques. Ils peuvent ainsi décrire la situation ou l'état du système selon ces données.
  Cette catégorie comprend des systèmes de surveillance, de compréhension du langage et d'analyse d'images.

- Les systèmes de prédiction : ils déduisent les conséquences problables à partir d'une situation donnée.
  Cette catégorie inclut des systèmes de prévisions météorologiques, de prédictions démographiques ou de traffic et d'estimations de récoltes.

- Les systèmes de diagnostic : ils déduisent à partir d'observations, les disfonctionnements du système.
  Ces systèmes peuvent être entre autres, des diagnostics médicaux, électroniques ou mécaniques.

- Les systèmes de design : ils développent des configurations d'objets qui satisfont les contraintes du problème de design.
  De tels problèmes peuvent être le tracé d'un circuit ou le modèle d'un bâtiment à construire.

- Les systèmes de planning : ils prescrivent des actions.
  Cette catégorie reprend des problèmes de programmation automatique comme les robots, les communications, les itinéraires et les problèmes de planification militaire.

- Les systèmes de monitoring : ils comparent les observations issues du comportement du système aux caractéristiques cruciales à son bon fonctionnement.
  Il existe beaucoup de systèmes de monitoring pour les centrales nucléaires, le traffic aérien et la fiscalité.

Un système expert est composé
- d'une interface
- d'une base de connaissances acquise auprès des experts d'un domaine spécifique
- d'un système cognitif qui contient un mécanisme de résolution de problèmes utilisant la base de connaissances et les données du problème. Il s'agit donc d'un mécanisme de raisonnement caractéristique à la façon de raisonner des experts, et qui permet au système de fonctionner comme un assistant intelligent du praticien.

Nous pouvons donner quelques exemples de systèmes experts.

- MYCIN : il s'agit d'un système développé à partir de 1977 par Shortliffe et ses collègues [BUC84]. Il est destiné à établir un diagnostic et une thérapie dans le domaine des maladies infectieuses du sang, d'urine ou autre. Son travail consiste à récolter des renseignements, puis à donner un diagnostic sur la bactérie présente chez le patient, et à conseiller enfin une thérapie appropriée.

- DENDRAL : ce système expert fut développé principalement par Buchanan, Mithchell et Feigenbaum et a débuté à la fin des années 60. Sa fonction est d'interpréter des données provenant d'un spectographe de masse. Il analyse également la résonance magnétique nucléaire et des données chimiques pour inférer les structures possibles d'un composant inconnu.

- HEARSAY-II fut développé durant les années 70. Il s'agit d'un système de compréhension du langage.

Le but de notre mémoire est de contribuer au développement d'un système expert conseil en mise en peinture.

La plupart des personnes désireuses de peindre un support sont confrontées au problème du choix des peintures à appliquer. Ce problème se révèle de plus en plus difficile en raison de la variété

toujours plus vaste des produits sur le marché. Un conseil en peinture s'avère donc utile. Ce conseil est habituellement donné par des experts en peinture. Afin de seconder ceux-ci, ou de les remplacer lorsqu'ils n'existent pas, un système expert fut élaboré par Karine Declerq et Jacqueline Parache dans le cadre de leur mémoire (1985-1986).

L'objectif de ce système expert etait de conseiller la suite de traitements que l'utilisateur doit appliquer sur le support qu'il souhaite peindre. Cette solution doit tenir compte des exigences du client et des caractéristiques du support. L'utilisateur donne la description du support dans son état actuel et dans l'état désiré après application de la solution que le système expert lui proposera. La construction d'une telle solution consiste à sélectionner des traitements qui permettent de transformer la surface du support de l'état actuel à l'état désiré. La sélection d'un traitement doit respecter des règles de compatibilité technique. Dans ce système expert, seuls les supports métalliques étaient pris en considération et son implémentation fournit tous les systèmes de peinture possibles, en fonction uniquement de critères techniques, relatifs aux spécificités de la peinture.

Afin de nous familiariser avec ce système expert, nous lui avons ajouté la possibilité de prendre en compte les supports en bois.

De plus, le nombre de systèmes de peinture conseillés pouvant être assez important pour un même problème considéré, l'utilisateur risque de se trouver face à plusieurs solutions techniquement viables, mais parmi lesquelles il lui est impossible de faire un choix. Notre objectif est de l'aider dans un tel choix en intégrant au système expert des critères commerciaux. Ces critères reprennent les politiques de vente menées au sein de la firme Trimetal, les aspects économiques d'une peinture (son prix, son temps de séchage, ...) et les aspects qui dépendent de l'appréciation de l'utilisateur (l'utilisateur peut avoir une préférence pour les anciens ou les nouveaux produits, pour les traitements à appliquer avec un pinceau ou un rouleau, ...). Ces critères permettent de départager plusieurs solutions techniquement correctes et de conseiller la (les) solution(s) la (les) plus avantageuse(s) pour la firme et pour l'utilisateur.

Un dernier objectif de notre travail est de permettre à l'utilisateur d'obtenir la justification du conseil donné. En effet, par définition, tout système expert doit être capable d'exposer le raisonnement suivi et les principes appliqués pour aboutir à la solution.

La manière d'atteindre ces trois objectifs sera détaillée dans le présent travail selon la structure suivante.

Le deuxième chapitre est destiné à rappeler ce qui a été réalisé dans le mémoire de K. Declercq et de J. Parache. Nous exposerons quel est le problème traité, quels sont les concepts utilisés, quelles sont les démarches et la modélisation adoptées pour résoudre ce problème.

Le troisième chapitre traite de l'extension du système Conseil aux supports en bois. Nous verrons que le modèle sous-jacent à la mise en peinture des supports métalliques peut être repris, moyennant quelques adaptations, afin de tenir compte des spécificités du bois. Cette constatation n'a pu être faite que grâce à de long entretiens avec les experts et à une étude approfondie des caractéristiques du bois.

Le quatrième chapitre est consacré à la prise en compte de critères commerciaux. Il débute par une répertorisation de ces différents critères. Une telle classification relève des politiques commerciales appliquées par l'entreprise, des décisions du marketing et des préférences de l'utilisateur. L'ajout de tels critères nous mènera à étudier quel type de méthode de recherche paraît le plus adéquat pour effectuer la construction d'un système de peinture, le choix ne se référant maintenant plus seulement à des critères techniques. Une analyse des divers types de méthodes de recherche utilisées en intelligence artificielle nous permettra de montrer que la recherche adoptée dans le mémoire de l'année 1986 n'est plus appropiée au problème considéré. L'approche pour laquelle nous avons opté se base sur des principes de recherche heuristique.

Le cinquième chapitre propose une technique d'explication permettant de décrire à l'utilisateur les stratégies et les choix que le sytème expert a appliqués et de les justifier. Une première partie détaille ce qu'est un processus d'explication, ce en quoi il est utile et les critères de conception qu'il doit respecter. La deuxième partie expose les différents types d'explications nécessaires aux différents types d'utilisateurs désireux d'obtenir un conseil en peinture. Il est en effet important de connaître quelles sont les questions que chaque utilisateur est susceptible de poser et quelles sont les réponses qui doivent y être apportées. Différentes approches pour modéliser un processus d'explication sont exposées dans un troisième point. Le dernier point décrit la méthode que nous proposons. Le processus que nous avons élaboré est le fruit de notre recherche, nous verrons en effet que les méthodes classiques ne semblent pas adéquates au problème spécifique de la mise en peinture.

En conclusion, nous présentons une évaluation du travail réalisé. Nous évoquons quels sont les objectifs que nous n'avons pu atteindre et quelles en sont les raisons. Cette dernière partie présente également quels sont les travaux ultérieurs qui pourraient constituer une suite logique de ce travail. Les problèmes que nous avons rencontrés sont aussi abordés dans ce point.

Enfin, la fin de ce mémoire est composée des annexes dans lesquelles se trouvent les interviews que nous avons menés auprès des experts, les caractéristiques techniques des produits sur lesquels le système expert se base, et enfin le code de la base de connaissances et du programme PROLOG.

II. UN PREMIER SYSTEME CONSEIL : RAPPELS SUR UNE REALISATION ANTERIEURE

II.1. DESCRIPTION DU PROBLEME

Le système Conseil élaboré par K. Declercq et J. Parache dans le cadre de leur mémoire [DEC86], a pour but de conseiller le client désireux de peindre un support, dans le choix des peintures à appliquer. Il permet de résoudre le problème suivant : à partir de deux états, l'un initial et l'autre final, d'un support à peindre, donner la séquence des traitements que le support dans son état initial doit subir pour atteindre son état final. La séquence de traitements vide est une solution acceptable. L'état initial est l'état du support au moment de la requête, avant tout traitement, reprenant notamment ses dégradations et ses traitements antérieurs. L'état terminal est l'état du support après application de la séquence des traitements résultat du système Conseil, c'est-à-dire l'état dans lequel le client souhaite amener le support. Ces états doivent être décrits selon des critères communs de façon à pouvoir les comparer aisément. Le travail de K. Declercq et J. Parache a été limité aux supports métalliques.

II.1.1. Les trois étapes du conseil en peinture

Le conseil en peinture se déroule en trois étapes principales.

La première étape est l'introduction d'une description des états initial et final du support par l'utilisateur. Il s'agit ici de faire connaître au système les caractéristiques de ces deux états.

La deuxième étape est la construction de la suite de traitements à partir de ces deux états. Une telle suite est appelée SYSTEME DE PEINTURE. Elle doit respecter des contraintes techniques propres à la peinture et doit répondre à la demande de l'utilisateur. Une recherche est donc à faire parmi tous les traitements possibles de la gamme Trimetal. Selon leurs caractéristiques (l'aspect qu'ils donnent, ...), ils peuvent être un élément de solution pour le problème de l'utilisateur. Les caractéristiques d'un produit sont décrites au point II.1.2. La recherche se fait à partir de l'état final du support pour arriver régressivement à l'état initial grâce à la construction d'états intermédiaires. Un état intermédiaire est considéré comme état du support après application d'une suite de traitements. Il s'agit d'une recherche régressive (backward).

Cette étape peut être formulée de la façon décrite ci-dessous.

Etant donné une suite de traitements partiellement construite, et l'état intermédiaire correspondant à l'étape courante de la recherche (état intermédiaure "cible") deux types de problèmes se posent :

- il faut d'abord déterminer un nouveau traitement permettant d'atteindre un nouvel état intermédiaire. Ce choix se fait à l'issue de deux phases

    1. une comparaison entre l'état intermédiaire "cible" et l'état du support après application d'un traitement candidat. Les caractéristiques de ce dernier doivent être compatibles avec l'état "cible".

    2. une vérification de la compatibilité chimique entre un traitement candidat ayant franchi la phase 1 avec succès et le dernier traitement choisi dans la séquence construite jusqu'alors.

- une fois déterminé un traitement candidat d'après les phases 1 et 2, il faut mettre à jour l'état intermédiaire comme étant le nouvel état cible.

Ce processus itératif se termine lorsque le nouvel état intermédiaire obtenu n'est rien d'autre que l'état initial.

La troisième étape est la sortie à l'écran de la séquence résultat dérivée par la deuxième étape.

II.1.2. L'état du support et les traitements

Le problème est régi par deux concepts. Le premier est celui d'ETAT DU SUPPPORT, qui est défini comme l'état d'un objet susceptible de subir un traitement. Le deuxième concept est celui du TRAITEMENT, qui intervient comme élément dans la séquence résultat, et est défini soit comme l'application d'un produit de la gamme TRIMETAL, soit comme une opération de préparation.

Un état du support est caractérisé par les attributs décrits ci-dessous :

- nature : type de matériau utilisé pour la fabrication du support.
  Exemple : acier, galvanisé, cuivre, bois.

- forme : situation dans l'espace de la surface à peindre ou forme générale de l'objet.
  Exemple : horizontale ou non, angulaire, tubulaire.

- environnement : conditions dépendant du milieu environnant le support (entre autres atmosphériques et chimiques).
  Exemple : marin, mural, intérieur aggressif.

- état de surface : appréciation de l'état du support à peindre par rapport à sa recouvrabilité.
  Exemple : corrodé, recouvert d'un produit en bon état, sale, non-poussiéreux.

- aspect : type d'aspect au sens utilisé en peinture.
  Exemple : brillant, filmogène, lasure.

- résistances : facteurs agressifs auxquels le support résiste.
  Exemple : ambiance agressive, chaleur.

- protections : dégradations auxquelles le support reste insensible.
  Exemple : la corrosion.

- traitements déjà appliqués : si le support n'est pas nu, liste des traitements que le support a déjà subis.
  Exemple : Plomoferrine N, Trisatin, ponçage.

- caractéristiques spéciales : facteurs exceptionnels dont il faut tenir compte, mais qui n'entrent dans aucune autre catégorie.
  Exemple : antidérapant, non-toxique.

Un traitement est caractérisé par les attributs décrits ci-dessous :

- composition chimique : composants chimiques utilisés lors de la fabrication d'un produit, et renseignements dépendants de la composition.
  Exemple : solvant, PVC.

- conditions d'application : conditions matérielles (c'est-à-dire outillage) et atmosphériques à remplir pour l'application du traitement; il s'agit de renseignements portant sur la façon d'appliquer le traitement et d'exigences portant sur le support avant application du traitement.
  Exemple : brosse, T° ambiante, support adhérent.

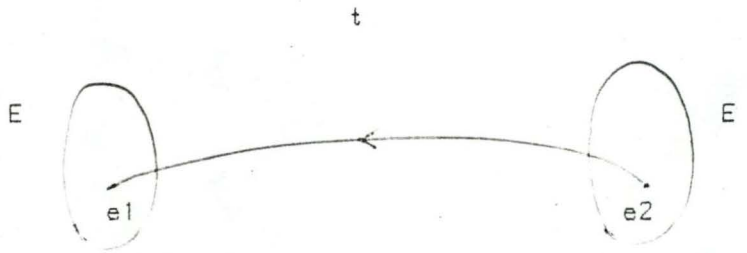
- résistances : facteurs agressifs auxquels le traitement résiste.
  Exemple : atmosphère industrielle, solvants.

- protections : dégradations contre lesquelles le traitement protège le support.
  Exemple : corrosion, infiltration d'eau.

- caractéristiques spéciales : qualités que le traitement apporte au support et qui n'entrent dans aucune autre catégorie.
  Exemple : antidérapant, non-toxique.

Remarquons que les caractéristiques de tout support et de tout traitement reprises ci-dessus sont purement TECHNIQUES.

Les concepts d'ETAT DU SUPPORT et de TRAITEMENT sont liés par une relation qui peut se schématiser comme suit :



avec e1, e2, deux états intermédiaires du support.

Tout TRAITEMENT t est défini par une fonction t : E -> E, où E est l'ensemble des états possibles. Cette fonction exprime le fait que la mise en peinture d'un SUPPORT dans un ETAT donné (e1) à l'aide d'un TRAITEMENT t modifie cet ETAT pour donner un nouvel ETAT (e2).

Cette fonction est définie ssi l'application du traitement t n'est pas incompatible avec les valeurs des caractéristiques de e1 : nature, forme, environnement, état de surface, aspect, résistances, protections, caractéristiques spéciales et traitements déjà subis. L'effet de toute application de t à un état de support e1 est d'attribuer aux caractéristiques aspect, protections, résistances et caractéristiques spéciales de e1 les valeurs des caractéristiques correspondantes de t. Cet effet donne naissance à l'état e2.

Les conditions d'application d'un traitement à un état particulier sont déterminées par

- des règles dépendant de la composition chimique de ce traitement et du dernier traitement appliqué,
- des règles concernant les résistances, les protections et les aspects de ce traitement et de l'état de surface, et
- des règles portant sur le type de support.

Ce deuxième type de règles détermine, entre autres choses, si le traitement apporte un élément de réponse au problème, c'est-à-dire s'il satisfait au moins une caractéristique de l'état du support.

Exemple.

Si pour l'état final du support, la caractéristique "aspect" a pour valeur "brillant, lisse", la caractéristique "résistances" a pour valeur "chaleur" et les caractéristiques "protections" et "état de surface" ont les valeurs par défaut, alors un traitement ne sera sélectionné que si son aspect est brillant ou lisse ou si il assure une résistance à la chaleur.

Ces règles constituent les principes contraignant la construction d'un système de peinture. Elles forment avec les caractéristiques techniques du support et du traitement les critères techniques.

II.2. RECHERCHE D'UN SYSTEME DE PEINTURE SELON UNE APPROCHE "ESPACE D'ETATS"

- MODELE

Rappelons qu'une modélisation toute naturelle pour représenter le problème de la mise en peinture est la modélisation par "espace d'états".[NIL71]

Dans cette modélisation, il y a deux concepts importants : le concept d'état et celui d'opérateur. Le problème peut être décrit par un quadruplet <S, I, F, OP> où

S est un ensemble d'états;
I est un ensemble d'états initiaux représentant les données du problème et leurs propriétés (I S);
F est un ensemble d'états finals représentant les solutions du problème et leurs propriétés (F S);
OP est un ensemble d'opérateurs de transformation d'états

(OP : S -> S).

Dès lors, résoudre un problème revient à déterminer une suite d'états intermédiaires Si tels que

So I, Sn F,
Si = OPk (Si-1) où OPk est applicable à Si-1.

La solution au problème est constituée de la suite d'opérateurs OPij

avec Sn = OPin (OPin-1 (... (OPi1 (So))...) où So I et Sn F.

La méthode de résolution consiste à générer de nouvelles descriptions, Si, à partir d'anciennes, Si-1, par application d'un opérateur OPk et ensuite à vérifier si Si a les propriétés décrites par Sn : c'est le principe classique du "GENERATE AND TEST".

- APPLICATION

Pour le conseil en peinture, la représentation des éléments de l'espace d'états se fait grâce au formalisme des vecteurs.

Les états, en l'occurrence, les descriptions du support, sont représentés par des agrégats dont les différentes composantes sont les caractéristiques techniques définies auparavant; plus précisément, l'état d'un support est représenté par un VECTEUR D'ETAT, dont chaque caractéristique est un couple du type (ATTRIBUT,VALEUR), où ATTRIBUT

caractéristique est un couple du type (ATTRIBUT,VALEUR), où ATTRIBUT est le nom d'un critère technique et VALEUR est la valeur de ce critère pour le support considéré.

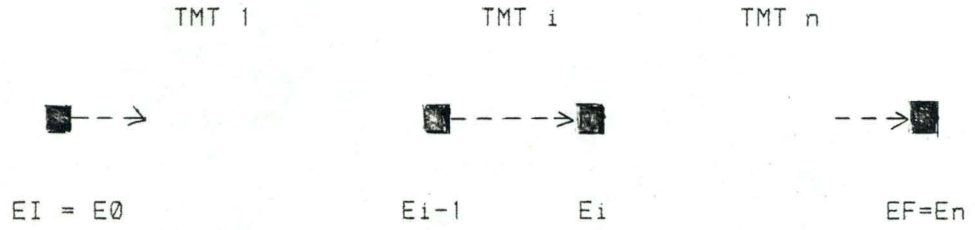
Les opérateurs, dans notre cas, les traitements, sont représentés par des règles de réécriture de forme générale

TRAITEMENT : PRECONDITION ----> POSTCONDITION

où PRECONDITION et POSTCONDITION caractérisent des états de l'espace et donc des VECTEURS D'ETATS. PRECONDITION caractérise l'état du support avant application du traitement, et POSTCONDITION caractérise l'état du support après application du traitement.

L'utilisation de variables dans PRECONDITION permet de décrire tous les états possibles de l'espace et leurs instanciations permettent d'obtenir la description d'un état particulier.

La solution recherchée, pour un état initial et un état final donnés, peut dès lors être représentée par le schéma suivant :



EF = TMTn (TMTn-1 (... (TMT1 (EI))...))

où ■ représente un support, caractérisé par des VECTEURS D'ETATS Ei (0 <= i <= n) et où ——→ représente l'application d'un TRAITEMENT. Le TRAITEMENT TMTi modofie l'état du support en l'amenant de l'état Ei-1 à l'état Ei.

En réalité, la recherche est régressive. Elle part de l'état final EF du support et applique régressivement les traitements TMTi aux états Ei pour générer les états intermédiaires Ei-1 et arriver à l'état initial EI.

L'espace d'états est l'ensemble de toutes les descriptions possibles d'un support et de toutes les transitions possibles entre ces descriptions. Une recherche dans cet ensemble permet alors de trouver une solution (si elle existe) à un problème donné.

III. EXTENSION DU SYSTEME CONSEIL AUX SUPPORTS EN BOIS

Le modèle sous-jacent à la mise en peinture sur des supports métalliques peut être repris pour des supports en bois puisque la recherche d'une suite de traitements pour peindre des supports de différentes natures est le même.

En effet, quelle que soit la nature du support, un traitement sera sélectionné s'il apporte un élément de réponse au problème et si ses caractéristiques sont compatibles avec l'état du support. Il suffit d'adapter et de compléter la terminologie employée aux spécificités du bois. Les jeux de valeurs des caractéristiques sont évidemment différents (on ne trouve pas de champignons sur le métal, alors que pour le bois, il est nécessaire de le traiter). Eventuellement, il sera peut-être nécessaire d'ajouter des nouvelles caractéristiques propres au bois. Toutefois, le processus de raisonnement est le même.

Nous pouvons dès à présent souligner la différence principale entre le métal et le bois : contrairement au métal, le bois est un matériau organique et vivant. Le point crucial du bois est l'humidité, tandis que celui du métal est la corrosion.

Les points développés ci-dessous décrivent les modifications à apporter à la description du support et des traitements applicables aux boiseries. Ces modifications sont des ajouts (par rapport à ce qui a été fait pour les métaux [DEC86]) de valeurs pour chaque caractéristique. Nous évoquons également les contraintes chimiques que les traitements pour bois doivent respecter. La conclusion développe comment le système étendu aux bois à été construit, évalue le caractère extensif du modèle et cite succintement quelles furent les difficultés rencontrées.

III.1. DESCRIPTION DU SUPPORT

Le modèle sur lequel nous nous basons étant le même que celui développé pour le métal, nous allons reprendre les différents attributs caractérisant un support dans un état déterminé et en spécifier les valeurs possibles pour le bois. Ces valeurs seront ajoutées au modèle de [DEC86].

Rappelons que dans [DEC86], les valeurs possibles pour chaque attribut du vecteur d'état ont été classées selon deux manières non-exclusives. La manière selon laquelle une valeur est répertoriée indique les règles à appliquer en ce qui concerne la compatibilité entre un traitement et un état du support.

Ce classement est décrit ci-dessous [DEC86].

1. Quand il y a une gradation dans les exigences représentées par chaque valeur, elles sont données dans l'ordre croissant.

Exemple.

Pour les supports métalliques, l'attribut "environnement" peut prendre les valeurs "rural", "urbain" et "industriel". Si le support est situé dans un environnement industriel, il doit être recouvert d'un traitement plus résistant que si le support se trouvait dans un environnement rural ou urbain. L'environnement industriel est donc plus exigeant que les environnements urbain et rural. De même, l'environnement urbain est plus exigeant que l'environnement rural. L'ordre de ces valeurs est donc rural, urbain, industriel.

Une des règles de compatibilité qui découle de cet ordre est la suivante : un traitement adapté à l'environnement industriel convient pour un support situé dans un environnement urbain ou rural, puisque ceux-ci sont moins exigeants. Par contre, un traitement adapté à l'environnement rural ne convient pas pour un support situé dans un environnement urbain ou industriel, puisque ces deux environnements sont trop exigeant pour le traitement considéré.

2. Un système de classes a été établi pour répertorier les différentes valeurs.

Classe a) : valeurs à supprimer, c'est-à-dire qui n'apparaissent plus dans Ei si elles sont spécifiées dans la postcondition du traitement TRTi; sinon, elles sont maintenues dans Ei (Ei est l'état obtenu après application du traitement TRTi). Ces valeurs devront tôt ou tard être satisfaites.

Exemple : si la valeur de l'attribut "résistances" est "corrosion" dans l'état final du support, cela signifie que l'utilisateur désire une protection contre la corrosion du support. Il faudra donc sélectionner tôt ou tard un traitement assurant cette protection. C'est la raison pour laquelle "corrosion" fait partie de la classe des valeurs à supprimer.

Classe b) : valeurs à remplacer, c'est-à-dire qui doivent apparaître nécessairement dans la postcondition du traitement TRTi si elles sont spécifiées dans Ei. Dès que ces valeurs apparaissent dans un état du support, elles doivent être immédiatement satisfaites.

Exemple : si la valeur de l'attribut "aspect" est "brillant" dans l'état final du support, cela signifie que l'utilisateur désire que la dernière couche donne un aspect brillant. Il est donc nécessaire de satisfaire cette caractéristique immédiatement. La recherche étant régressive, le premier produit sélectionné est la couche de finition. Si cette carctéristique n'est pas satisfaite tout de suite

mais bien plus tard (après la sélection du premier traitement), le traitement de finition risque de ne pas être brillant et dans ce cas, les désirs de l'utilisateur ne seront pas exaucés.

Classe c) : valeurs permanentes, c'est-à-dire qui doivent être spécifiées dans chacun des Ei (1 <= i <= n).

Exemple : si la valeur de la composante "aspect" est "translucide" dans l'état final du support, cela signifie que toutes les couches doivent être translucides. La sélection d'un traitement opaque annulerait l'aspect translucide de toute autre couche. L'aspect translucide doit donc être satisfait à chaque sélection d'un produit.

Nous allons à présent citer toutes les valeurs des attributs décrivant l'état d'un support en bois.

1. Nature du bois (obligatoire et non répétitive)

Dans notre travail, nous ne considérons pas les bois industriels. Nous étudierons les bois en bâtiment que nous pouvons subdiviser en catégories :

- les bois durs et les bois sensibles, la durabilite d'un bois se définissant par sa capacité à résister aux micros-organismes. Pour les bois sensibles, des traitements spécifiques devront être réalisés afin de mieux les protéger contre les bactéries, les champignons, etc.

- les bois feuillus et les bois résineux, les premiers posant un problème de mauvaise pénétration de la peinture et donc d'adhérence, les derniers posant un problème d'exsudation de sève.

- les bois européens et les bois tropicaux.

Nous pouvons également distinguer les bois selon leur variété : sapin, hêtre, chêne, meranti, merbau, afzélia, teck ... .

Cependant, les bois utilisés en menuiserie sont de bonne qualité, c'est-à-dire qu'ils n'exsudent pas de sève, qu'ils ne nécessitent pas de traitement fait par des spécialistes ... . Un système de peinture, quelle que soit sa qualité, ne permettra pas d'améliorer la qualité du bois.

La nature spécifique d'un support en bois n'intervient donc pas dans le choix du traitement. Il n'est dès lors pas nécessaire de distinguer les catégories décrites ci-dessus.

Les valeurs possibles décrivant la nature du bois se réduisent donc à :

- bois.

2. Forme du support (facultatif et non-répétitif)

La forme du support en bois n'a pas d'importance puisqu'elle n'implique pas de traitement ou de produit particulier. Cet attribut n'est donc pas considéré pour les supports en bois.

3. Environnement du support (obligatoire et non-répétitif)

Seule la distinction entre environnement intérieur et extérieur sera faite, la distinction entre environnement rural, urbain et industriel ne se justifiant que lorsque le critère de corrosion doit être pris en considération.

Les produits prévus pour l'extérieur contiennent plus de protection, notamment contre les intempéries. Dès lors, tout produit prévu pour l'extérieur peut s'appliquer à l'intérieur, le contraire n'étant pas vrai.

A l'extérieur, les produits transparents et non teintés ou mats sont à éviter car ils laissent passer les filtres ultra-violets qui dégradent le bois. Ces traitements ne sont pas assez durables à l'extérieur.

Pour obtenir un bon résultat, la mise en peinture devra se faire dans un milieu où la température ambiante est supérieure à 5°C et inférieure à 35°C et où l'humidité est inférieure à 75%.

Les valeurs possibles sont les suivantes :
- extérieur
- intérieur.

4. Caractéristiques spéciales (facultatif et répétitif)

Valeur par défaut : indifférent; signifie qu'aucune caractéristique spéciale n'est exigée.

Ces caractéristiques feront l'objet d'un questionnaire supplémentaire lorsqu'un choix entre plusieurs solutions possibles devra être fait.

Les valeurs possibles sont les suivantes :

- étendue du support : grande ou petite. Dans cette composante, nous spécifierons si la surface à peindre est grande ou petite. Cet aspect est important car selon l'étendue du support, la peinture s'applique plus ou moins bien.
- séchage rapide
- facilité d'application (pas d'outillage spécial, application aisée de la peinture ...)
- sans coulure
- peu salissant
- aspect décoratif : généralement, l'aspect décoratif n'a pas beaucoup d'importance à l'extérieur, sur une charpente, ... alors qu'il en a pour certains supports à l'intérieur.

Ces cinq dernières valeurs sont en fait des caractéristiques du support dont la valeur est celle du traitement appliqué à l'état du support. N'oublions pas que l'effet de l'application de tout traitement t à un état du support e est d'attribuer aux attributs aspect, protections, résistances et caractéristiques spéciales de e les valeurs des attributs correspondants à t.

5. Etat de surface (obligatoire, décomposable et répétitif)

Tout support devant être mis en peinture doit d'abord être mis à nu ou tout au moins être en bon état. C'est ce que nous appelons un support recouvrable. Si le bois est neuf, seul un dépoussiérage sera nécessaire. Si le bois est déjà traité, il devra subir plusieurs traitements avant la mise en peinture proprement dite. Dans ce cas, on dira que le support est au départ dégradé.

Il existe deux types de dégradations :

a) dégradations propres au bois : noeuds, exsudation de sève, poche de résine, ... Dans ce cas, il s'agit de bois de mauvaise qualité qui requiert un traitement fait par des spécialistes. L'unique application de produits Trimetal ne sera pas suffisante.

b) dégradations autres :

* dégradations chimiques et biologiques

- les micro-organismes
  -) insectes, pourriture, champignons.
  Si ces dégradations sont assez importantes, elles demandent l'intervention de spécialistes et le système ne pourra pas donner de solution.

  -) bleuissement : cette dégradation est provoquée par des bactéries s'infiltrant dans le bois. Elle peut être fortement atténuée par un lavage à l'eau de javel. Il

est cependant impossible de la faire disparaître complètement.

Ces dégradations sont dues à un taux d'humidité trop important. Il sera donc capital de maintenir le taux d'humidité de tout bois (dégradé ou pas) inférieur à celui permettant la survie des micro-organismes.

- les rayons ultra-violets (U.V.) qui provoquent un grisaillement ou une décoloration. Malheureusement, il n'existe pas de produit dans la gamme Trimetal qui supprime ce grisaillement.

* dégradations mécaniques

  - fissuration : il n'existe pas de produit Trimetal pour ce type de dégradation
  - peinture écaillée
  - cloquage : cette dégradation est due au fait que la peinture utilisée empêche l'humidité contenue dans le bois de sortir.

* défauts, aspérités : un enduit sera nécessaire afin d'égaliser les surfaces.
* humidité : on considère que le bois est sec si le pourcentage d'eau qu'il contient est inférieur à 18% pour les boiseries extérieures, et inférieur à 10% pour les boiseries intérieures et les parquets. Si ces pourcentages sont dépassés, le bois doit être séché.
* arêtes vives : il est préférable de poncer les arêtes afin de les arrondir et donc d'empêcher la peinture de fuir les arêtes et d'éviter une érosion trop rapide à ces endroits.
* poussière
* graisse (résine)
* saleté.

Les valeurs possibles de l'attribut "état de surface" sont donc les suivantes :

classe a)

* dégradé : - noeuds
  - exsudation de sève
  - fissuration
  - porosité
  - biologique : non, faible ou fort
  - bleuissement
  - U.V.
  - peinture écaillée
  - cloquage : non, faible ou fort
  - défauts
  - humidité
  - arêtes vives

- poussière
- graisse
- propreté

* recouvrable : bois en bon état qui demande à être rafraîchi.

6. Aspects (facultatif et répétitif)

Valeur par défaut : indifférent; signifie que l'aspect peut être quelconque sans restriction.

Les valeurs possibles sont les suivantes :

classe a)

- filmogène

classe b)

- opaque
- brillant
- satiné
- mat

classe c)

- translucide
- lisse
- non-filmogène
- vitrifié.

7. Résistances du support (facultatif, décomposable et répétitif)

Valeur par défaut : aucune; signifie qu'aucun type de résistance n'est assurée.

Les valeurs possibles sont les suivantes :

classe a)

- abrasion mécanique

* abrasion légère : résistance moyenne aux griffes
* abrasion moyenne : bonne résistance aux griffes
* abrasion forte : bonne résistance aux chocs (généralement pour escaliers et parquets)

- abrasion chimique

  * eau : résistance au lavage à l'eau claire (généralement, tous les produits résistent à un lavage léger si l'eau ne stagne pas sur le support pendant un long moment)

  * lavage moyen : résistance au lavage à l'eau savonneuse
  * lavage : résistance aux détergeants ménagers
  (* intempéries : tous les produits prévus pour l'extérieur résistent aux intempéries. L'environnement étant obligatoirement spécifié, cette valeur est inutile.)

- stagnation d'eau

(- contact de la terre : voir applications particulières).

Il est aisé de comprendre que pour le bois, il n'existe pas de produit qui le protègerait de manière plus ou moins efficace contre le feu.

8. Protections du support (facultatif et répétitif)

Valeur par défaut : aucune; signifie qu'aucun type de protection n'est assuré.

Les valeurs possibles sont les suivantes :

classe a)

- bleuissement
- pourriture, champignons
- insectes
- U.V.

9. Applications particulières (facultatif et non-répétitif)

Valeur par défaut : aucune; signifie que le support ne fait pas partie d'une application particulière, c'est-à-dire qu'il n'est ni une charpente, ni une menuiserie extérieure ayant un contact avec la terre.

Cet attribut est nécessaire, car pour chacune de ces applications un seul produit de la gamme Trimetal convient. Ces produits donnent un seul aspect (l'utilisateur n'a donc pas le choix quant à cette caractéristique) et toutes les résistances et protections que de telles applications nécessitent.

Les valeurs possibles sont les suivantes :

classe a)

- charpente
- menuiseries de jardin (menuiseries extérieures ayant un contact avec la terre).

10. Nom du dernier traitement

III.2. LES TRAITEMENTS

Nous abordons dans un premier temps ce qui constitue la description d'un traitement pour bois et nous verrons les quelques adaptations à effectuer pour la sélection d'un traitement pour bois par rapport au modèle de [DEC86]. Ensuite, nous expliquons ce qu'il est nécessaire d'ajouter à cette description pour que la distinction entre traitement pour bois et pour métaux puisse être faite. Enfin, nous détaillons les traitements particuliers que sont les préparations.

1. La description des traitements pour bois est identique à celle des traitements pour métaux. Elle comprend donc les attributs suivants :

- nom
- précondition
- postcondition
- composition : toute peinture est composée de liants, de pigments et de solvants. La composition détermine la quantité de chacun de ces éléments contenue dans le produit considéré.

Le modèle et les règles sous-jacents à la sélection d'un traitement pour bois sont identiques à ceux permettant la sélection d'un traitement pour métal. Le modèle sera simplement complété de règles spécifiques aux bois et des règles concernant l'attribut "applications particulières" mentionnée ci-dessous.

La règle de sélection d'un traitement en ce qui concerne les applications particulières est la suivante : les valeurs permanentes de l'attribut "applications particulières" de l'état courant doivent être contenues dans la postcondition du traitement, si ce traitement n'est pas une préparation.

La règle de passage au nouveau vecteur d'état en ce qui concerne l'attribut "applications particulières" (attribut que nous avons dû ajouté au modèle de [DEC86]) est la suivante : les valeurs permanentes et les autres valeurs de l'attribut "applications particulières" qui ne sont pas dans la postcondition du traitement sélectionné sont reprises dans l'attribut "applications particulières" du nouveau vecteur d'état.

Dans l'état actuel des choses, toutes les valeurs de l 'attribut "applications particulières" sont permanentes. Cependant, le caractère adaptable du modèle nous oblige à prévoir l'ajout d'autres valeurs.

2. L'attribut "nature" d'un traitement

Les traitements destinés aux métaux ne conviennent pas nécessairement pour les supports en bois. En effet, certains de ces traitements (Zincano, Zincanoe, Washprimer, ...) contiennent des types de pigments qui lors de l'application provoquent une dégradation du bois. D'autre part, certains traitements pour bois n'adhèrent pas au métal, de sorte que ces traitements ne peuvent être appliqués sur un support métallique.

Tout produit appliqué sur bois doit être perméable à la vapeur d'eau afin que l'humidité contenue dans le bois puisse s'échapper et qu'il n'y ait pas de cloquage. Il doit également résister à un minimum d'infiltration d'eau, l'eau amenant des bactéries qui endommagent le bois.

Ces traitements se différencient grâce à la valeur "bois" de l'attribut "nature" du vecteur d'état.

3. Préparations

Les préparations permettent la mise en peinture du bois. Nous distinguons 11 préparations propres aux bois.

- L'appel aux spécialistes : l'intervention d'un spécialiste est nécessaire lorsque le bois contient des noeuds, des fissurations, lorsqu'il est poreux, lorsqu'il exsude de la sève ou lorsqu'il est fortement attaqué par des insectes, ...
  Cette préparation doit se faire en tout premier lieu.

- Le grattage : cette préparation doit être effectuée lorsque le bois a subi un cloquage faible ou lorsque la peinture sur le bois est faiblement écaillée.

- Le décapage qui doit se faire lorsque le bois a subi un cloquage fort ou lorsque la peinture sur le bois est fortement écaillée.

- Le séchage qui doit se faire lorsque le support est humide.

- Le ponçage des arêtes : cette préparation doit être réalisée pour éviter que la peinture ne fuie les arêtes vives, ce qui entrainerait une dégradation plus rapide à ces endroits.

- Le dégraissage qui doit se faire lorsqu'il y a de la graisse sur la surface du bois.

- Le dépoussiérage qui doit être effectué lorsqu'il y a de la poussière sur la surface du support ou après un grattage ou un ponçage.

- Le lavage qui doit être réalisé lorsque la surface du bois est sale.

- Le lavage à l'eau de javel qui doit être effectué si la surface du bois a une coloration bleue.

- L'application d'une couche de Xylamon Combi qui doit être effectuée si le bois est faiblement attaqué par des insectes, ...

- L'enduisage qui doit être effectué si la surface du bois comprend des défauts (qui risquent d'être apparents si la finition a un aspect brillant).

III.3. LES CONTRAINTES CHIMIQUES

L'application d'une suite de traitements successifs doit respecter certaines contraintes chimiques, sous peine de voir les couches se détruire l'une l'autre. Ces contraintes sont exactement les mêmes pour tout type de support (métal, bois, maçonnerie ...) car elles sont relatives à la composition des peintures, c'est-à-dire à leurs liants, leurs solvants et leurs pigments. Il existe deux contraintes importantes : celle se basant sur le principe de la thermoplasticité et celle se basant sur le concept de P.V.C..

III.3.1. La thermoplasticité

La thermoplasticité doit croître ou rester constante en partant du primaire vers la finition, c'est-à-dire qu'un produit duroplaste (dur) ne peut recouvrir un produit thermoplaste (élastique). En effet, ce dernier suit les déformations subies par le support en bois alors que la couche supérieure plus dure, reste figée. Dans ce cas, les mouvements de la couche inférieure entrainent des craquelures.

III.3.2. Le P.V.C. (Pigmentation Volume Concentration)

La formule du PVC d'une peinture s'exprime de la manière suivante :

$$P.V.C. = \frac{\text{volume des pigments}}{\text{volume des pigments + volume des solvants}}$$

Si le produit considéré est un vernis, il ne contient pas de pigments et son PVC est nul. Une couche d'un tel produit (PVC faible) constitue une surface bien lisse, non poreuse et forme donc un vrai film. Ceci justifie le caractère brillant d'une peinture (la valeur du PVC est plus ou moins inversément proportionnelle au caractère brillant) et l'utilisation de peintures à faible PVC comme couche de finition, le film empêchant une érosion trop rapide.

Si le produit considéré contient très peu de liants, son PVC sera élevé. Une couche de ce produit est poreuse et ne forme pas un vrai film, ce qui entraine une érosion très forte.

Nous pouvons en déduire les contraintes suivantes :

- le PVC doit décroître du primaire vers la finition

- le caractère brillant doit croître du primaire vers la finition pour une raison d'adhérence, la brillance d'une couche empêchant l'accrochage de la couche supérieure

- seuls les produits ayant un PVC inférieur à 0.40 peuvent être en contact avec l'air car une peinture trop poreuse s'érode beaucoup trop vite. Un tel système n'est pas durable.

III.4. CONCLUSION

Le système étendu prenant en compte les supports en bois a été obtenu à partir du modèle traitant les supports métalliques en y apportant les modificatons suivantes :

- la base de données est complétée de tous les traitements pour bois
- les valeurs des attributs de la description de l'état du support et des traitements sont complétées par les valeurs décrites ci-dessus et qui sont spécifiques au bois
- l'attribut "applications particulières" est ajoutée aux attributs décrivant l'état du support et les règles relatives à ce nouvel attribut sont insérées au programme.

Grâce à ces ajouts et à ces modifications, le système Conseil peut traiter les supports en bois. Le modèle de base développé dans [DEC86] est donc très facilement extensible à tout type de support.

Les principales difficultés rencontrées sont d'ordre humain. Ce problème est abordé dans la conclusion générale de ce mémoire.

IV. PRISE EN COMPTE DES CRITERES COMMERCIAUX

L'objectif de ce chapitre est d'intégrer des critères commerciaux dans le choix d'un système de peinture. Une brève introduction permet de comprendre pourquoi la prise en considération de tels critères est nécessaire.

Le deuxième point détaille quels sont les critères commerciaux en vigueur chez Trimetal.

La manière dont ces critères vont intervenir dans la recherche d'un système de peinture est abordée dans les points suivants.

IV.1. INTRODUCTION

Le programme CONSEIL issu du mémoire réalisé par Karin Declercq et Jacqueline Parache ([DEC86]) donne TOUS les systèmes de peinture TECHNIQUEMENT corrects selon l'état initial et l'état final du support. Selon cette première version, le choix d'un système de peinture se définit uniquement en fonction de ces critères techniques.

Dès lors, le nombre de systèmes techniquement corrects pour un même problème est généralement trop important, ce qui risque d'amener la confusion auprès de l'utilisateur. Ceci est dû à la multitude de produits existants dans la gamme Trimetal Paint, parmi lesquels beaucoup de produits possèdent des caractéristiques qui se recouvrent.

Exemple : pour peindre un support en bois au moyen d'une peinture opaque, lisse et brillante, on peut utiliser du Stellapaint, du Permacryl ou de la Permaline.

Afin de réduire le nombre de solutions et de guider le choix de l'utilisateur vers la solution la plus favorable, il est nécessaire d'ajouter d'autres critères, tels les critères commerciaux. Un conseil en peinture se fait effectivement sur base des critères techniques ET commerciaux.

Le nombre de solutions à présenter à l'écran par le programme ne doit pas obligatoirement être égal à un, mais il dépend de l'utilisation qui en sera faite selon le type d'utilisateur :

- le client d'une grande surface qui, ignorant les spécifités de la peinture ne peut être confronté seul à plusieurs solutions. C'est la raison pour laquelle une seule solution lui sera proposée. Eventuellement, afin de lui laisser une certaine liberté, le système pourrait lui présenter deux solutions chacune complétée des caractéristiques qui les distinguent, pour aider le client à effectuer un choix. Mais son aptitude de jugement étant très faible dans le domaine de la peinture, il est préférable de lui présenter le moins possible de solutions.

- les représentants et le personnel de l'entreprise conseillant les clients, qui n'auront accès qu'à un ou deux systèmes de peinture, afin d'éviter une discordance entre les conseils.

- les chimistes qui désirent TOUS les systèmes techniquement corrects en fonction du problème posé. A partir de toutes ces solutions, ils peuvent effectuer certaines recherches, notamment la détection des utilisations non conventionnelles des produits et le repérage des cas pour lesquels il n'existe pas de système de peinture, ce qui pourrait déclencher la mise au point de nouveaux produits. Pour ce type d'utilisateur, le système devra offrir la possibilité de ne tenir compte que des critères techniques.

Le nombre de solutions désirées par l'utilisateur est fixé lors de la description de son problème. Cependant, il est possible qu'il n'existe pas autant de solutions techniquement correctes que le désire l'utilisateur.

IV.2. LES POLITIQUES COMMERCIALES

Les politiques commerciales appliquées dépendent de la société dans laquelle le conseil en peinture est effectué. L'analyse de l'organisation de la firme Trimetal permet de connaître quelles sont les sociétés qui la composent. De là, nous pouvons déduire quel est l'ensemble des politiques à prendre en considération. Nous verrons également comment elles seront formalisées et si elles sont toujours d'application en fonction du type de peinture.

IV.2.1. L'organisation de Trimetal

Les politiques commerciales diffèrent dans chaque société, il est donc important de connaître comment s'organise la société Trimetal.

L'entreprise Trimetal est composée de 6 sociétés :

- Trimetal Belgique
- Trimetal Nederland
- Trimetal France
- Roll S.A.
- Trimetal Italie
- Mäder Bénélux

Ces sociétés se distinguent par les réseaux de distribution par lesquels elles passent et par les secteurs de vente qu'elles touchent :

1) Trimetal Belgique vend par le circuit de distribution long : producteur - grossiste - détaillant ou applicateur.

Les secteurs de vente ainsi touchés sont :

- le "do it yourself"
- le professionnel
- les grands travaux (entreprises de peinture, Ministères, ...)

Chacun de ces secteurs comprend des gammes de produits communs d'une part, et distincts d'autre part.

Exemple : il n'y a pas de qualité différente selon que le client appartient au grand public ou s'il est professionnel, mais le profesionnel a accès à plus de produits de qualité, étant donné la limitation technique du "do it yourself".

2) Trimetal Nederland a les mêmes caractéristiques que Trimetal Belgique, mais développe une tendance à distinguer le professionnel du "do it yourself" et donc à vendre des qualités différentes selon le type de client.

3) Trimetal France et Roll S.A. fournissent par le circuit de distribution long mais ne s'adressent qu'aux professionnels.

4) Mäder Bénélux vend par le circuit de distribution court, c'est-à-dire qu'il fournit directement aux revendeurs et aux peintres (professionnels et industries).

Les sociétes Trimetal fournissent en grande partie aux grossistes, tandis que Mäder Bénélux fournit principalement aux grandes surfaces.

Nous verrons de quelle manière les distinctions entre ces sociétés se traduisent dans le choix d'une peinture.

IV.2.2. Types de politiques commerciales

Les facteurs extra-techniques que nous pouvons qualifier de "commerciaux" et qu'utilisent les experts afin de conseiller le client sur la suite des traitements à appliquer, peuvent être de deux types :

- des facteurs permettant d'exclure certaines solutions (filtres). Ces facteurs seront nommés "régles commerciales".

- des facteurs permettant de guider le choix de l'utilisateur parmi les solutions restantes. Ces facteurs seront nommés "règles d'agrément".

IV.2.2.1. Les règles commerciales

- En raison des subdivisions de Trimetal, les produits vendus par une société ne peuvent être vendus par une autre. Par conséquent, notre système ne propose pour chaque problème posé, que des produits d'une seule société et non pas des produits de deux sociétés différentes.

- Le système ne propose que des produits standards, par opposition aux produits spéciaux qui ne se trouvent pas au tarif.

- Le but principal des commerciaux est de vendre et de satisfaire le client. Il est donc important que le système de peinture proposé ait tous ses composants sur le lieu de vente considéré, c'est-à-dire la droguerie, la grande surface ou chez le grossiste. Il est inutile de proposer une solution que l'on ne peut trouver sur place. Nous introduirons alors la notion de "produits présents" et de "produits absents" sur le marché, chaque produit étant amovible dans ces deux groupes. Cette notion inclut également les variations de stock.

IV.2.2.2. Les règles d'agrément

Ces règles d'agrément sont divisées selon 2 types de critères.

1. Les critères extra-techniques sans question : il s'agit de critères à REPONSE PREDEFINIE, c'est-à-dire des critères pour lesquels il n'est pas indispensable de connaître l'avis de l'utilisateur, cet avis étant généralement le même quelle que soit la personne demandant un conseil.

2. Les critères extra-techniques avec question : il s'agit de critères DE PREFERENCE, exigeant l'appréciation de l'utilisateur.

1. Les critères extra-techniques sans question sont les suivants :

   * le prix/m2 du produit : le client préfère utiliser la suite de traitements la moins chère possible. En outre, pour l'entreprise, il est préférable de proposer des produits dont le prix est le plus proche possible de celui de la concurrence.

   * la simplicité du système : le système de peinture doit comprendre le moins possible de produits différents afin d'éviter l'utilisation non complète d'une multitude de boîtes de peinture.

   * la composition du produit : l'utilisateur éprouve une certaine réticence à employer des produits qui ne sont pas monocomposants, c'est-à-dire pour lesquels il faut mélanger soi-même les composants.

   * les promotions de produits

* la base du produit : la base (acqueuse ou solvants) peut dépendre de l'application. En effet, les produits à base solvants dégagent une odeur plus forte et sont donc à éviter notamment dans une chambre d'enfants, les produits à base acqueuse sont mieux adaptés pour revêtir un endroit exposé à une forte humidité, ...etc.

  En outre, les sytèmes comprenant des produits de même base sont préférables aux systèmes dont les produits sont de bases distinctes.

* le temps de séchage et par conséquent le temps de traitement que l'utilisateur souhaite, doit être minimum. Il existe pour un même produit, deux temps de séchage : un temps minimum que l'on considère si ce produit est la couche de finition et un temps maximum si ce produit est utilisé comme une couche qui doit être recouverte.

* la facilité d'application.

Remarque : cette liste ne retient pas le taux de toxicité. En effet, aucun des produits applicables à l'intérieur n'est toxique. Seuls les produits prévus pour l'extérieur peuvent contenir des matières toxiques, mais l'application de telles peintures se faisant en dehors d'une habitation, ne présente aucun danger pour l'être humain.

Le critère de toxicité n'intervient donc pas dans le choix d'une peinture.

Pour chacun de ces facteurs, le système agira de lui-même dans le sens indiqué ci-dessus, c'est-à-dire qu'il conseillera, dans la mesure du possible, un système de peinture de prix minimum, comprenant peu de produits, ne comprenant pas de produits à mélanger ou de base différentes, comprenant des produits en promotion, aisés à appliquer et dont le temps de séchage est faible.

Le système doit également tenir compte du type de client :

* le particulier qui demande un bon rapport qualité/prix

* le professionnel qui demande le prix le plus bas. Il est également important de tenir compte des habitudes du professionnel. Or, la tendance actuelle et la politique de Trimetal, est de passer des systèmes solvants aux systèmes acqueux, ce qui est contraire aux habitudes des peintres professionnels. Dans ce cas, il faudrait peut-être présenter les deux solutions, la solution à base solvants pour respecter ses habitudes et la solution à base d'eau afin de faire connaître les nouveaux produits. A ce niveau, nous pouvons constater le gros inconvénient de l'ordinateur : par manque de

relations humaines, la probabilité de convaincre le client de l'efficacité d'un nouveau produit est beaucoup plus faible que lorsque le conseil est donné par téléphone. Par dialogue oral, les contacts sont plus aisés et ceux-ci permettent de mieux ressentir quels sont les préférences et les désirs du client.

2. Les critères extra-techniques avec question sont les suivants :

* la qualité désirée : selon le type d'application, la qualité supérieure ou de luxe n'est pas toujours requise.

  Exemple : - la remise à neuf d'un mobilier se trouvant dans un living

  - la protection d'un meuble se trouvant dans un grenier ou dans une cave.

  Dans le premier cas, il s'agit d'un travail "soigné" et la qualité du produit est très importante. Dans le deuxième cas, il s'agit d'un travail "courant" et une qualité moyenne est suffisante.

  Cette notion de travail courant ou soigné englobe la notion d'aspect décoratif désiré par l'utilisateur. Il est alors nécessaire de savoir si le support comprend des petites ou des grandes surfaces puisque selon l'étendue du support et la base du traitment, l'aspect est plus ou moins décoratif.

  Le niveau de qualité demandé a en outre une influence sur le prix du produit choisi en conséquence.

* la période de l'année : selon les conditions climatiques certains types de peintures sont mieux adaptés.

  Exemple : lorsqu'il fait fort humide, la peinture ne s'évapore pas très facilement et il est préférable d'utiliser un système à base acqueuse. La période que nous considérons trop humide, s'étend habituellement du mois d'octobre au mois d'avril.

* le mode d'application c'est-à-dire le matériel à utiliser.

* l'ancienneté du produit : généralement, un choix se pose entre un produit nouveau et un produit plus ancien. A ce niveau, il s'avère que certaines personnes sont réticentes aux nouveaux produits qui n'ont pas encore fait leurs preuves, tandis que d'autres sont prêtes à utiliser de nouveaux produits qui sont habituellement plus performants.

  Le terme "ancienneté" pourrait être remplacé par "notoriété" car c'est surtout en fonction de la réputation et de la publicité du produit que l'utilisateur fait son choix.

* les fonctions supplémentaires qu'offrent les systèmes de peinture candidats au problème posé par l'utilisateur, par rapport aux exigences de ce dernier. Ces fonctions supplémentaires reprennent entre autres des caractéristiques quasi obligatoires pour un système de peinture.

  Exemple : pour un support métallique, il est préférable de proposer un système avec anti-corrosion, même si le client ne demande pas cette protection.

  Ceci permettrait d'inciter le client à utiliser des systèmes de peinture de bonne qualité.

* système à base solvants ou acqueuse : bien que la tendance du marché soit un marché vers des produits sans solvant (en raison de sa toxicité), sans odeur, qui permettent un rinçage aisé des pinceaux, ... , les solutions les plus favorables au client ne sont pas toujours les solutions contenant de tels produits. C'est la raison pour laquelle il est important de connaître l'avis de l'utilisateur.

* odeur : il est important de savoir si le client veut absolument un système sans odeur, ou si cet aspect lui est indifférent. Si peu lui importe ce critère, on n'en tient pas compte dans la recherche.

L'ensemble de ces facteurs extra-techniques sont ceux que nous avons déduits des interviews effectués dans la firmè Trimetal. Ce sont les critères en fonction desquels l'expert aide le client à trouver un système de peinture répondant à son problème. Cependant, la personnalité de l'expert (ses préférences, ...), qui exerce une grande influence sur son jugement, n'a pu être prise en considération car cet aspect est impossible à formaliser. Elle se caractérise par différentes estimations des critères (selon la préférence de l'expert, il jugera qu'un produit est facile ou difficile à appliquer) et par un dosage différent de chacun d'eux dans l'évàluation d'un système de peinture. Nous considérons donc l'ordinateur comme un expert à part entière, avec sa propre personnalité et ses propres préférences.

La formalisation de tels concepts, si elle eut été possible, aurait été à l'encontre d'un de nos objectifs qui est l'uniformisation de la source d'information c'est-à-dire l'unicité du conseil. L'implémentation sur ordinateur du conseil en peinture permet en effet de donner toujours la méme solution pour un même problème.

IV.2.2.3. Critères guidant le choix d'un système de peinture

Toute suite de produits sélectionnée comme solution au problème posé, doit respecter des critères de sélection. L'ensemble des critères est donc ce qui détermine en quoi une suite de traitements est solution à un problème.

Il existe deux types de critères.

- Critères techniques : critères qui déterminent les conditions d'applicabilité d'un produit. Ce type de critère ne porte que sur le choix d'UN produit en fonction d'un état intermédiaire (qui détermine les caractéristiques qu'il reste à satisfaire), indépendamment des produits choisis auparavant.

  Il existe parmi les critères de ce type, un critère d'utilité. Grâce à celui-ci, un produit n'est choisi que s'il apporte quelque chose au problème, s'il permet de se rapprocher de la solution. Un traitement n'est choisi que s'il satisfait un des attributs de l'état intermédiaire. Dans ce cas, c'est un produit utile. Cependant, il ne s'agit pas d'un produit nécessaire car il existe généralement plusieurs produits différents satisfaisant les mêmes caractéristiques.

  Si le produit est utile, il doit encore respecter les autres critères d'applicabilité ; il doit convenir avec les attributs de l'état intermédiaire : il doit être de même nature, doit assurer les protections nécessaires à l'environnement du support, ...

- Critères économiques (extra-techniques) : critères qui fixent le choix à faire entre plusieurs solutions candidates (non nécessairement complètes) pour que la solution ainsi retenue soit la plus adéquate aux désirs de l'utilisateur. Ils portent donc sur une SUITE de produits déjà sélectionnés, et tiennent compte de tous les traitements appartenant à cette suite. Il s'agit de critères d'optimalité. Ils reprennent les politiques commerciales.

Il existe un ordre de satisfaction de ces critères : il correspond à l'ordre en fonction duquel l'ordinateur procède pour vérifier qu'un ensemble de traitements pourrait être une solution.

Pour construire une solution, un traitement est sélectionné si en premier lieu, il respecte les critères techniques et si la nouvelle solution ainsi formée respecte les critères économiques.

Par cet ordre :
- un produit satisfaisant un critère d'applicabilité satisfait également les critères d'applicabilité précédents
- une suite de produits satisfaisant un critère d'optimalité satisfait également les critères d'optimalité précédents
- les produits d'une suite satisfaisant un critère d'optimalité satisfont tous les critères d'applicabilité.

IV.2.3. Formalisation des critères commerciaux

Nous avons vu au chapitre III que tout traitement est décrit par 4 attributs : son nom, sa nature, sa précondition et sa postcondition (contenant les critères techniques).

Les critères commerciaux sont formalisés par une attribut supplémentaire. Cet attribut est nommé "vecteur d'effets de bord" et constitue une liste de couples contenant le nom du critère commercial et la valeur de ce critère pour le produit considéré.

Exemple.

Si le traitement TRT a un prix au mètre carré A, des temps de séchage (minimum et maximum) B et C, une composition D, une base E, une qualité M et si la valeur du critère de promotion est F, du critère de facilité est G, du critère de stock est H, du critère odeur est I, du critère ancienneté est J, du critère mode d'application rouleau est K et du critère mode d'application pistolet est L, le vecteur effet de bord de TRT est le suivant :

```
effetsdebord (TRT, [[pm, A],
                    [tsech, [B, C]],
                    [comp, D],
                    [base, E],
                    [prom, F],
                    [fac, G],
                    [stock, H],
                    [odeur, I],
                    [anci, J],
                    [maplir, K],
                    [maplip, L],
                    [qual, M]]).
```

avec TRT, le nom du traitement,
pm, signifiant prix au mètre carré,
tsech, signifiant temps de séchage,
comp, signifiant composition,
prom, signifiant promotion,
fac, signifiant facilité,
anci, signifiant acienneté,
maplir, signifiant mode d'application par rouleau,
maplip, signifiant mode d'application par pistolet,
qual, signifiant qualité,
A, un réel,
B, C, des entiers,
D valant "mono" signifiant mono-composant
ou "bi" signifiant bi-composant,
E valant "acqueuse" ou "solvants",
F, G, H, I, J, K, L valant "oui" ou "non",
M, valant "luxe" ou "bonne".

IV.2.4. Applicabilité des critères commerciaux aux différents types de peintures

Les critères commerciaux cités au point IV.2.2., sont applicables à presque tous les produits que nous envisageons, c'est-à-dire à presque tous les produits standards. De tels critères sont valables notamment aussi bien pour des peintures standards s'appliquant sur du bois que sur du métal.

La non-applicabilité d'un critère pourrait se présenter dans le cas où des produits spécifiques sont pris en considération ou pour quelques rares produits standards destinés à une utilisation particulière.

Exemple : un enduit (reboucheur de pores) n'a pas de base.

Si le critère i n'est pas applicable au produit j, la valeur du critère i dans le vecteur d'effets de bord du produit j est affecté à une valeur conventionnelle (valeur nulle) pour exprimer cette incompatibilité.

IV.3. RAPPELS SUR LES METHODES DE RECHERCHE DANS L'ESPACE D'ETATS

Comme nous l'avons vu, le problème de la construction d'un système de peinture est modélisé par une recherche dans un espace d'états. Nous rappelons brièvement ici quelques éléments concernant les techniques classiques de recherche dans une approche "espace d'états" [NIL71].

Cette modélisation permet de structurer le problème de manière à le décomposer en sous-problèmes plus aisés : rechercher le produit à appliquer pour passer d'un état à un autre.

Cette représentation peut être schématisée par un graphe dans lequel les noeuds représentent les états de l'espace et les arcs représentent les opérateurs permettant de passer d'un état à un autre.

Le problème à résoudre détermine dans ce graphe un noeud de départ et un noeud but à atteindre. La recherche dans l'espace d'états définit un chemin c'est-à-dire une suite d'arcs, ayant pour origine le noeud de départ et pour extrémité le noeud but. La solution est cette suite d'arcs, d'actions à effectuer pour atteindre le noeud but à partir du noeud de départ. Mais généralement, il existe différents chemins joignant deux noeuds distincts et différentes méthodes pour découvrir ces chemins. De sorte que plusieurs stratégies de recherche sont possibles.

Notons que généralement, à tout arc peut être associé un coût représentant le coût de l'application de l'opérateur correspondant à cet arc. Dans certains cas, nous pouvons être amenés à rechercher un chemin de coût minimal, le coût d'un chemin étant la somme des coûts de chaque arc composant le chemin considéré.

Pour toute méthode :

- la première étape de la recherche se base sur le noeud de départ

- chaque étape intermédiaire génère les successeurs du noeud choisi à l'étape précédente et choisit parmi les noeuds déjà générés, le noeud pour lequel le chemin qui permet de l'atteindre est de coût minimal

- la dernière étape de la recherche procède à la reconstitution du chemin composé des noeuds et des arcs sélectionnés.

La génération des successeurs d'un noeud est appelée développement de ce noeud.

La sélection du noeud à développer se fait en fonction du coût du chemin qui lui est associé.

Selon la définition du coût d'un opérateur, la statégie de recherche ainsi obtenue diffère.

IV.3.1. Méthodes exhaustives de recherche

Les méthodes exhaustives génèrent TOUS les noeuds de l'arbre mais selon la méthode, l'ordre dans lequel ils sont générés varie.

Afin d'exposer les différentes méthodes exhaustives, procédons à quelques définitions.

Définition de la profondeur d'un noeud :

- la profondeur d'un noeud initial = 0
- la profondeur d'un noeud quelconque est la profondeur de son parent augmentée de un.

Définition d'un niveau k : le niveau k est l'ensemble des noeuds de profondeur k.

Les différents types de recherches exhaustives sont les suivants.

* Recherche en largeur d'abord (BREADTH FIRST)

  Tout noeud de profondeur k est developpé avant tout noeud de profondeur k+1. Elle développe donc les noeuds du niveau 1, puis ceux du niveau 2, ...

* Recherche en profondeur d'abord (DEPTH FIRST)

Tout noeud de profondeur a ses descendants développés avant le noeud suivant de profondeur k. Elle développe donc le premier noeud de niveau 1 et en génère ses descendants, puis le premier noeud de niveau 2 et en génère ses descendants, ...

* Recherche en profondeur limitée d'abord

Cette recherche est un compromis entre les recherches en largeur d'abord et en profondeur d'abord.

Dans cette stratégie, l'arbre est découpé en tranches, chaque tranche étant composée d'un même nombre k de niveaux. On appelle ancêtre d'une tranche tout noeud dont le parent appartient à la tranche précédente.

La recherche se fait :

- en profondeur d'abord dans une tranche : tout noeud de profondeur k a tous ses descendants de cette même tranche développés avant tout noeud suivant de profondeur k.
- en largeur d'abord entre tranches : tout ancêtre est développé avant tout ancêtre de la tranche suivante.

IV.3.2. Inconvénients de la recherche exhaustive

- Une telle recherche entraine une explosion combinatoire du nombre de noeuds à visiter.

- Ce type de recherche ne tient pas compte des données particulières du problème qui pourraient déterminer quels noeuds il est préférable de visiter afin d'atteindre au plus vite une solution, éventuellement optimale.

Nous sommes donc obligés de visiter tous les noeuds jusqu'à ce que le noeud but se présente, ce qui entraine d'une part la certitude de trouver la solution (éventuellement optimale) si elle existe, mais d'autre part, une inefficacité en temps et en place.

Beaucoup de grands problèmes combinatoires (comprenant un grand nombre de données et pemettant de nombreuses possibilités) ne peuvent être résolus par une analyse exhaustive des cas. Pour de tels problèmes, il est plus efficace d'utiliser une heuristique afin de guider la recherche.

IV.3.3. Méthodes heuristiques de recherche

Ces méthodes constituent des techniques d'optimisation qui ont pour but de réduire le nombre de noeuds à développer pour arriver à un noeud but. Le principe est d'exploiter simultanément un nombre variable de chemins et de poursuivre ceux qui semblent être prometteurs.

Il existe différents types de méthodes heuristiques.

A. Heuristiques AVEC fonction d'évaluation

On se définit une fonction d'évaluation :

f : S --> R où S est l'espace d'états
R est l'ensemble des réels

f(s) est l'évaluation numérique de la promesse de l'état s d'être sur le chemin de coût minimal allant de l'état initial à l'état final (noeud but).

Cette fonction d'évaluation représente une mesure de la "distance" ou de la "différence" d'un noeud par rapport au noeud but.

Une méthode heuristique avec fonction d'évaluation consiste à développer le noeud dont l'évaluation est la plus faible, c'est-à-dire du noeud le plus prometteur, le plus proche du noeud but. C'est celui qui a le plus de chance d'être sur le chemin de coût minimal allant de l'état initial à l'état final.

Pour faire une recherche heuristique, nous devons disposer de plusieurs noeuds à la fois afin de pouvoir comparer leur évaluation.

Cette recherche permet de trouver en un minimum de temps et avec un minimum de place mémoire, un chemin allant de l'état initial à l'état final.

Cependant, le choix d'une "bonne" fonction d'évaluation est crucial.

En effet, la détermination de cette fonction dépend de deux facteurs antagonistes :

1) la réduction maximale du nombre de noeuds générés
2) la garantie de trouver un chemin de coût minimal allant de l'état initial à l'état final.

Le conflit entre ces deux facteurs se traduit de la manière suivante :

- plus on veut réduire l'effort de recherche (facteur 1), plus il y a de risques de sous-estimer la promesse réelle de certains noeuds et donc plus la garantie de trouver un chemin de coût minimal s'amenuise (facteur 2).

- plus on veut être sûr de trouver un chemin de coût minimal (facteur 2), plus on risque de surestimer la promesse et donc plus le nombre de noeuds générés risque d'être élevé (facteur 1).

Il faut également considérer un autre aspect tout aussi important : plus la fonction d'évaluation est puissante et plus le calcul de sa valeur pour un noeud risque d'être coûteux.

La détermination d'une fonction d'évaluation devrait donc respecter un compromis entre la réduction de l'effort de recherche et la certitude de trouver un chemin de coût minimal afin d'obtenir un algorithme optimal (qui développe le moins de noeuds possibles) et admissible (qui garantit de trouver un chemin de coût minimal s'il existe).

B. Heuristique SANS fonction d'évaluation [ERM]

Ce type d'heuristique n'utilise pas de mesure numérique, mais différentes stratégies guidant la recherche.

La base de cette méthode est de construire un plan global qui comprend plusieurs stratégies, pour avancer vers l'état but en réduisant l'ensemble des chemins possibles de l'espace de recherche.

La méthode poursuit une stratégie jusqu'à ce qu'elle ne soit plus applicable ou qu'elle ne génère plus que des transitions illégales. A ce moment, un backtracking est effectué et une autre stratégie est choisie.

Cette méthode pourrait également admettre une création dynamique de stratégies et/ou utiliser des méta-stratégies afin de guider le choix entre ces différentes stratégies. La stratégie appliquée dépendrait du problème particulier qui est soumis au système, mais pourrait dépendre également de l'expérience du système.

Beaucoup de chercheurs en intelligence artificielle pensent en effet, qu'une qualité importante d'un comportement intelligent, est la capacité d'améliorer ses performances grâce à son expérience. C'est le concept crucial du "learning", ou apprentissage, où l'on procède notamment par généralisations et par analogies.

IV.3.4. Inconvénients de la recherche heuristique

- Toute méthode heuristique acceptable doit faire un compromis entre les critères d'admissibilité et d'optimalité.

- Selon le problème à résoudre, d'énormes difficultés peuvent se poser au niveau de la détermination de la fonction d'évaluation et des stratégies à utiliser.

IV.3.5. LE CHOIX D'UNE BONNE REPRESENTATION DE L'ESPACE D'ETATS

Les détails de la construction d'une bonne représentation de l'espace d'états et la recherche à travers celui-ci, dépendent du problème et de la personne qui le résoud. Si la tâche est simple, le processus de construction d'un espace d'états est triviale. Le problème est facilement assimilé à la connaissance de la personne et une stratégie est rapidement générée et exécutée.

Par contre, certains problèmes comme "La tour de Hanoï" et "Les missionnaires et les Cannibales" sont des exemples où un humain peut avoir des problèmes à construire une bonne représentation de l'espace d'états et une stratégie qui soient pertinents. Le temps pour résoudre des exemples de ce type, est bien sûr plus long par rapport au temps d'exécution de tâches simples.

La difficulté de résolution d'un problème augmente selon trois facteurs :

1) le système ne peut disposer de données précises ou de connaissances qui se calculent sans risque d'erreur.

2) lorsque les données changent dynamiquement, le système doit accélérer son raisonnement, baser certaines décisions sur des prévisions du futur, et revoir ses décisions quand une nouvelle donnée nie une supposition antérieure.

3) plus il y a de possibilités à considérer, plus la résolution est difficile. Il existe beaucoup d'applications où il est malaisé de quantifier la taille de l'espace d'états et de trouver d'autres formulations de celui-ci, qui simplifieraient autant que possible le problème.

IV.4. ALTERNATIVES POUR LA RECHERCHE D'UN SYSTEME DE PEINTURE

Le logiciel développé sur base des critères exclusivement techniques donne tous les systèmes de peinture techniquement corrects selon une recherche exhaustive en profondeur d'abord [DEC86]. Le choix d'une telle recherche trouve sa justification dans l'utilisation du langage Prolog dont l'interpréteur travaille en profondeur d'abord. L'ordre des systèmes solutions est fixé arbitrairement selon l'ordre des produits dans la base de connaissances.

Pour inclure les critères commerciaux dans le choix d'un système de peinture, plusieurs démarches sont possibles.

IV.4.1. Recherches exhaustives

Une première idée est de partir du logiciel développé sur base des critères techniques et de ne tenir compte des critères commerciaux que lorsque toutes les solutions techniquement correctes ont été trouvées.

Le système travaille en deux étapes :

- la recherche de toutes les solutions techniquement correctes
- la prise en considération des critères commerciaux afin de choisir quelle est, parmi ces solutions, la meilleure du point de vue commercial.

Il s'agit donc d'ajouter une analyse à ces solutions et de conserver intégralement le logiciel déjà développé pour les critères techniques.

IV.4.1.1. Prise en considération des critères commerciaux par la production de leurs valeurs à l'écran

Selon une telle approche, le programme donne TOUS les systèmes techniquement corrects complétés de commentaires concernant les caractéristiques propres à chacun de ces systèmes, en particulier ce que fait la solution en plus par rapport à ce que le client a demandé et les inconvénients de cette solution. Ces caractéristiques s'inspireraient des règles d'agrément.

Exemple : le client demande une protection contre le bleuissement du bois. Le système lui propose du GM Primer.

Pour cette solution, le système signalera que le GM Primer protège également contre les insectes et les moisissures. De plus il mentionnera le prix, le temps de séchage, ...

Avec ces informations supplémentaires, l'utilisateur pourra juger quel est le système lui convenant le mieux. Malheureusement, son jugement risque d'être ardu en raison du nombre d'informations données.

Un autre problème que soulève cette optique, est de savoir où s'arrêter dans l'énumération de ces caractéristiques. Une possibilité serait d'énumérer uniquement ce qui différencie les solutions proposées.

Avantage : * le système donnant toutes les solutions techniquement correctes, nous pouvons dire que le programme garantit de trouver la "meilleure solution".

Désavantages : * comme il doit chercher toutes les solutions, le programme est très lent

* l'utilisateur n'est pas aidé dans son jugement

* le choix final est réalisé par l'utilisateur, le même problème peut donc ne pas conduire à une solution unique.

Cette critique de la démarche, nous conduit vers une deuxième optique.

IV.4.1.2. Prise en considération des critères commerciaux dans le choix de la meilleure solution techniquement correcte

La deuxième optique est, à partir de toutes les solutions techniquement correctes préalablement calculées, de donner une et une seule solution, cette dernière étant la plus proche possible des souhaits de l'utilisateur. Eventuellement, les soutions pourraient être ordonnées selon les préférences de l'utilisateur. La réduction du nombre de solutions à une seule ou leur ordonnancement se ferait grâce à des questions supplémentaires se rapportant aux règles d'agrément (cfr IV.2.2.2).

Exemple : "Avez-vous un pistolet ?"

Si le client répond négativement, le système peut supprimer toutes les solutions nécessitant un pistolet pour peindre le support et revaloriser les solutions qui ne demandent pas de pistolet.

Le choix des questions posées se ferait selon l'idée de base suivante :
un système de peinture est déterminé selon différentes caractéristiques

A. les caractéristiques techniques (Ex : aspect, protection, ...).

B. les caractéristiques extra-techniques :
   * caractéristiques sans question c'est-à-dire les caractéristiques ne nécessitant pas de question (Ex : le client désire toujours un prix minimum)

   * caractéristiques avec question (Ex : l'odeur).

Etant donné deux systèmes de peinture techniquement corrects et répondant à un même problème, leurs caractéristiques techniques sont identiques, celles-ci étant entièrement déterminées par le problème de peinture considéré. Seules les caractéristiques extra-techniques peuvent être différentes. En effet, les solutions peuvent ne pas avoir le même prix, la même facilité d'application, ... Pour deux systèmes de peinture, il existe donc des caractéristiques extra-techniques ayant les mêmes valeurs et des caractéristiques extra-techniques ayant des valeurs distinctes. La préférence sera donnée au système dont les valeurs des caractéristiques extra-techniques sans question sont les plus favorables (cfr IV.2.2.2, point 1) : celui dont le prix et le temps de séchage sont les plus faibles, celui comprenant le moins de produit, ... L'ordonnancement ainsi obtenu doit également tenir compte des caractéritiques extra-techniques avec question non communes. En posant des questions se rapportant à ces caractéristiques, les réponses obtenues nous conduisent vers un nouvel ordonnancement ou vers le meilleur système de peinture si on désire une solution unique (le plus proche des souhaits de l'utilisateur).

Exemple .

Considérons les systèmes Permaline et Stellapaint. La Permaline est une peinture de très bonne qualité et se différencie ainsi de la Stellapaint. Le critère "qualité" est une caractéristique extra-technique qui distingue les deux systèmes. Le programme Conseil posera la question : "Désirez-vous une très bonne qualité ou une qualité satisfaisante ?". A partir de la réponse obtenue, le programme Conseil fera son choix.

Avantages : * l'utilisateur ne devra plus que répondre à des questions. Il n'est plus seul pour effectuer son choix, c'est le programme qui le conduit vers le meilleur système de peinture

* les caractéristiques sont sur un même pied d'égalité

* le programme garantit de trouver la meilleure solution. En effet, il supprime seulement les systèmes de peinture non désirés par l'utilisateur.

Désavantages : * le système Conseil est très lent car celui-ci doit chercher tous les systèmes techniquement corrects pour éventuellement en présenter seulement un, le

meilleur

* cette solution comporte le risque de poser un grand nombre de questions à l'utilisateur ce qui entraine un ennui et un désintèrêt vis-à-vis des réponses; celles-ci peuvent donc devenir de plus en plus imprécises. En effet, si à chaque système correspond une caractéristique extra-technique avec question différente, il faudra poser autant de questions qu'il n'y a de caractéristiques extra-techniques avec question.

IV.4.2. Recherches non exhaustives

Les deux précédents sytèmes présentent un désavantage commun : la lenteur d'exécution du programme, celui-ci devant attendre la fin de la recherche de tous les systèmes de peinture techniquement correct pour pouvoir choisir parmi ceux-ci, quel(s) est (sont) le(s) plus adapté(s) au problème de l'utilisateur. Cette recherche exhaustive entraine une importante perte de temps. L'idée est alors de conduire Conseil dans (et donc pendant) sa recherche. Il abandonnerait ainsi la recherche de nombreux systèmes corrects d'un point de vue technique mais peu satisfaisants du point de vue commercial, par rapport à d'autres solutions existantes. Il n'est pas nécessaire d'envisager de telles solutions. Le système peut donc les éliminer au fur et à mesure qu'il les rencontre. Cela permet de ne générer que les solutions qui sont techniquement correctes ET qui semblent être intéressantes du point de vue commercial.

Une recherche heuristique adaptée à notre application peut se matérialiser de différentes manières.

IV.4.2.1. Ordonnancement statique des produits dans la base de faits

Une première idée serait d'organiser les règles et faits Prolog de façon à ce que la première solution obtenue soit techniquement correcte et la plus favorable. Cela permet de stopper la recherche à cet endroit.

Les premiers produits de la base de données sont les premiers sélectionnés par l'interpréteur Prolog et doivent donc être les plus favorables.

Il s'agit donc d'ordonner les produits afin que le premier auquel l'interpréteur Prolog accède soit le "meilleur". Cet ordonnancement ne peut se faire que sur base des caractères extra-techniques sans question.

Exemple : produit A : prix 200Fr
temps de séchage 16 heures

produit B : prix 300Fr
temps de séchage 26 heures

produit C : prix 180Fr
temps de séchage 18 heures.

Si le critère prix est le critère le plus intéressant, l'ordre devient : C A B.

Avantage : * n'ayant aucun calcul à faire pendant l'exécution du programme, la réponse sera donnée rapidement.

Désavantages : * l'ordonnancement dépend de l'application et de l'échelle d'importance entre les critères extra-techniques. Or, selon l'utilisation, ces deux facteurs et donc l'ordonnancement varient de façon tout à fait aléatoire. Un réordonnancement de la base de données avant chaque exécution, constituerait un travail trop ardu.

* il n'est pas toujours possible de déterminer un ordre total.

* cette solution ne tient pas compte des caractéristiques extra-techniques avec question des systèmes de peinture.

* l'importance des critères n'est pas respectée si on ne modifie pas l'ordonnancement pour chacune des applications.

* l'ordonnancement des produits étant statique, le système n'est pas sûr de trouver la meilleure solution. En effet, les valeurs des critères extra-techniques peuvent varier au cours du temps (notamment le prix) et le premier produit de la base de faits risque de ne plus être le meilleur. Un réordonnancement des produits de la base de faits serait alors nécessaire.

Cette première méthode est fortement influencée par le langage Prolog. En effet, Prolog a un interpréteur travaillant en profondeur d'abord, ce qui entraine que le premier système de peinture sortant, le meilleur, doit se trouver au début de la base de données. On ne peut pas parler de recherche heuristique dans cette première méthode car pour une telle recherche, il faut visiter plusieurs noeuds en même temps. Or Prolog travaillant en profondeur visite un noeud puis va à son successeur. En raison de ce principe, la recherche du meilleur

noeud est réduite à sa plus simple expression car il n'y a qu'un noeud à choisir.

IV.4.2.2. Recherche heuristique

Une recherche heuristique est mieux adaptée à notre problème. En effet, pour un problème difficile, avec un important espace d'états, une résolution ne peut être efficace que si la recherche est efficace. L'espace d'états est tellement grand, qu'il ne peut être parcouru complètement. Le système doit donc utiliser des connaissances pour guider la recherche de façon à ce que relativement peu de points de l'espace d'états ne doivent être examinés avant de trouver la solution. Les informations dont dispose le système doivent être utilisées afin d'éliminer des sous-espaces entiers et simultanément de centrer la recherche sur des sous-espaces plus prometteurs. Nous abandonnons donc la statégie en profondeur d'abord utilisée dans [DEC86].

Rappelons que la recherche de systèmes de peinture est une recherche régressive (backward). En effet, elle se fait en commençant par l'état final (l'état désiré) du support pour arriver petit à petit à l'état initial de celui-ci. Le passage entre ces deux états se réalise par la construction d'un certain nombre d'états intermédiaires.

La suite de produits permettant de passer de l'état initial à l'état final du support, c'est-à-dire le système de peinture à soumettre à l'utilisateur, est celui obtenu en inversant les produits de la suite permettant de passer de l'état final à l'état initial du support.

Afin d'éviter toute confusion, une distinction importante est donc à faire : l'état initial du processus de recherche est l'état final du support et l'état final du processus de recherche est l'état initial du support.

La recherche à travers l'arbre est constituée du développement et de l'évaluation de certains noeuds. Dans la recherche d'un système de peinture, un noeud représente un état intermédiaire du support. Il représente également la suite de produits (traitements) qui a permis de l'atteindre.

Le développement d'un noeud se fait sur base des critères techniques et consiste dans notre cas à rechercher un nouveau traitement à la suite associée à ce noeud.

L'évaluation d'un noeud se fait sur base des critères extra-techniques sans question.

Comme nous l'avons vu précédemment, une recherche heuristique à partir d'un état initial, développe et considère PLUSIEURS états intermédiaires. Leur étude permet de déterminer l'état le plus intéressant et donc le sens dans lequel il est préférable de continuer

la recherche. Cependant, les états (et les solutions qui leur sont associées) qui n'ont pas été choisis ne sont que momentanément laissés en suspens. Ils seront de nouveau envisagés à l'étape suivante.

Le système construit donc progressivement plusieurs solutions simultanément. Chacune des suites de traitements se voit agrandir d'un traitement supplémentaire si le noeud associé à cette suite a l'évaluation la plus faible.

La recherche aboutit à un nombre des solutions qui peuvent être départagées grâce aux critères extra-techniques avec question. Les critères extra-techniques avec question ne sont pris en considération qu'APRES la recherche des meilleures solutions candidates. En effet, si une question est posée en cours de recherche, selon la réponse, nous risquons de ne plus considérer que des branches qui peuvent ne pas conduire à des solutions. De sorte que la recherche n'aboutirait à aucune solution uniquement à cause d'un critère extra-technique avec question.

Les concepts définis ci-dessous sont utiles à la compréhension de ce qui suit.

Quelques définitions

- Solutions candidate ou solution en cours : suite de traitements (système de peinture) construite à partir de l'état final du support, et son état courant, à savoir l'état intermédiaire obtenu après application (dans le sens état final vers état initial du support) de ces produits. Cette suite respecte les critères techniques. C'est donc une solution que le système est en train de construire et d'analyser pour savoir dans quelle mesure elle pourrait répondre au problème posé.

  En terme de graphe, il s'agit d'une suite d'arcs ou d'un chemin et du noeud terminal de celui-ci.

- Solution terminale ou solution complète : solution candidate dont l'état courant est l'état initial du support. Il s'agit donc d'un système de peinture permettant de passer de l'état final à l'état initial du support. Une telle solution résoud entièrement le problème de l'utilisateur du point de vue technique.

- Solution partielle : solution candidate dont l'état courant n'est pas l'état initial du support. En appliquant une telle suite de produits à l'état final du support, on ne peut pas atteindre l'état initial de celui-ci.

- Système-solution : solution terminale qui satisfait tous les critères économiques (ou extra-techniques). Une telle solution est par définition candidate et elle satisfait donc également les critères techniques. Seules ces solutions sont produites à l'écran pour en faire part à l'utilisateur.

Exemple : l'utilisateur veut peindre un support en bois dont la surface est nue, de manière à avoir un aspect brillant, une protection contre les U.V. et une résistance à l'eau.

Une solution complète serait une suite de traitements qui permet de satisfaire tous les désirs du client.

Une solution partielle serait une suite de traitements qui apporterait un aspect brillant, une résistance à l'eau mais qui ne protègerait pas contre les U.V.

- Solution "évaluée" : solution candidate pour laquelle la valeur de la fonction d'évaluation est calculée.

Algorithme de recherche

- Situation initiale

  * Le système développe l'état final du support, c'est-à-dire qu'il génère tous les états intermédiaires possibles grâce aux produits existants et en accord avec les critères techniques. Un ensemble de solutions candidates est obtenu. Il est décomposé en un ensemble de solutions partielles et un ensemble de solutions terminales.

  * Chaque solution est "évaluée" à l'aide d'une fonction d'évaluation construite sur base des critères économiques.

- Situation générale

  * Etant donné un ensemble de solutions partielles "évaluées" et un ensemble de solutions terminales "évaluées", le système développe la solution partielle dont la valeur de la fonction d'évaluation est la plus petite. L'enrichissement d'une solution partielle se fait par l'ajout d'un nouveau produit, selon les critères techniques.

  * Cette solution est supprimée de l'ensemble des solutions partielles "évaluées".

  * Les nouvelles solutions obtenues par ce développement sont "évaluées". Les terminales sont ajoutées à l'ensemble des solutions terminales "évaluées" et les partielles sont ajoutées à l'ensemble des solutions partielles "évaluées".

- Situation terminale

  Plusieurs cas peuvent provoquer l'arrêt de la recherche :
  * il n'existe plus de solution partielle à étendre.

  * le nombre de solutions terminales est supérieur ou égal au nombre

de systèmes de peinture désirés par l'utilisateur et il n'existe plus de solution partielle qui serait plus intéressante qu'au moins une solution terminale.

Dans le cas où ce nombre est supérieur, un choix est à faire à l'aide de questions posées à l'utilisateur, sur les caractéristiques extra-techniques avec question. Rappelons que ces caractéristiques dépendent entièrement des goûts du client et donc qu'il est nécessaire d'interroger celui-ci pour pouvoir les prendre en considération.

Les solutions finalement retenues, en fonction de la totalité des critères, sont les systèmes-solution.

Validation de l'algorithme de recherche

Nous allons démontrer la validité de cet algorithme de recherche, à savoir - qu'il donne au moins autant de solutions terminales que le désire l'utilisateur, si ces solutions existent (proposition 1)
- qu'il n'existe pas d'autres solutions plus intéressantes par rapport aux critères économiques (proposition 2)
- que l'algorithme s'arrête dans tous les cas, même lorsqu'il n'existe pas de solution techniquement correcte (proposition 3).

Ceci constitue en réalité, les spécifications de l'algorithme.

Pour démontrer ces trois propositions, la méthode de l'invariant semble toute indiquée. [GRI39]

L'invariant est le suivant :

on dispose d'un ensemble de pt solutions partielles "évaluées"
et on dispose d'un ensemble de tr solutions terminales "évaluées"

L'invariant est vrai lors de la première itération.

La valeur de la fonction d'évaluation pour l'état final du support est Ø et c'est la seule solution partielle "évaluée". A cette itération pt = 1 et tr = Ø.

L'invariant est vrai lors de deux itérations successives :

soient pt_i le nombre de solutions partielles "évaluées" lors de l'itération i,
tr_i le nombre de solutions terminales "évaluées" lors de l'itération i.

A la fin de l'itération i+1,

$$pt_{i+1} = pt_i - 1 + X$$
$$tr_{i+1} = tr_i + Y$$

avec X + Y le nombre de nouvelles solutions obtenues suite au développement du noeud choisi à l'itération et X + Y > 0
X étant le nombre de nouvelles solutions partielles,
Y étant le nombre de nouvelles solutions terminales.

Toutes les nouvelles solutions sont "évaluées".

Si l'algorithme se termine à l'itération n, il fournit des résultats corrects par rapport à la spécification décrite ci-dessus.

L'algorithme se termine pour différentes raisons :

- l'ensemble des solutions partielles est vide ($pt_n = \emptyset$).
  Toutes les solutions techniquement correctes ont été construites, elles sont contenues dans l'ensemble des solutions terminales (proposition 1)
- tr_n est supérieur ou égal au nombre de solutions souhaitées et les valeurs de la fonction d'évaluation pour toutes les solutions partielles sont supérieures à la valeur maximale de la fonction d'évaluation pour les solutions terminales. Il n'existe dons pas d'autres solutions plus intéressantes que les solutions terminales (proposition 2).

Le fait qu'à chaque itération l'algorithme choisit de développer la solution partielle la plus intéressante permet d'arriver plus vite au résultat.

Montrons à présent que l'algorithme se termine après un nombre fini d'itérations.

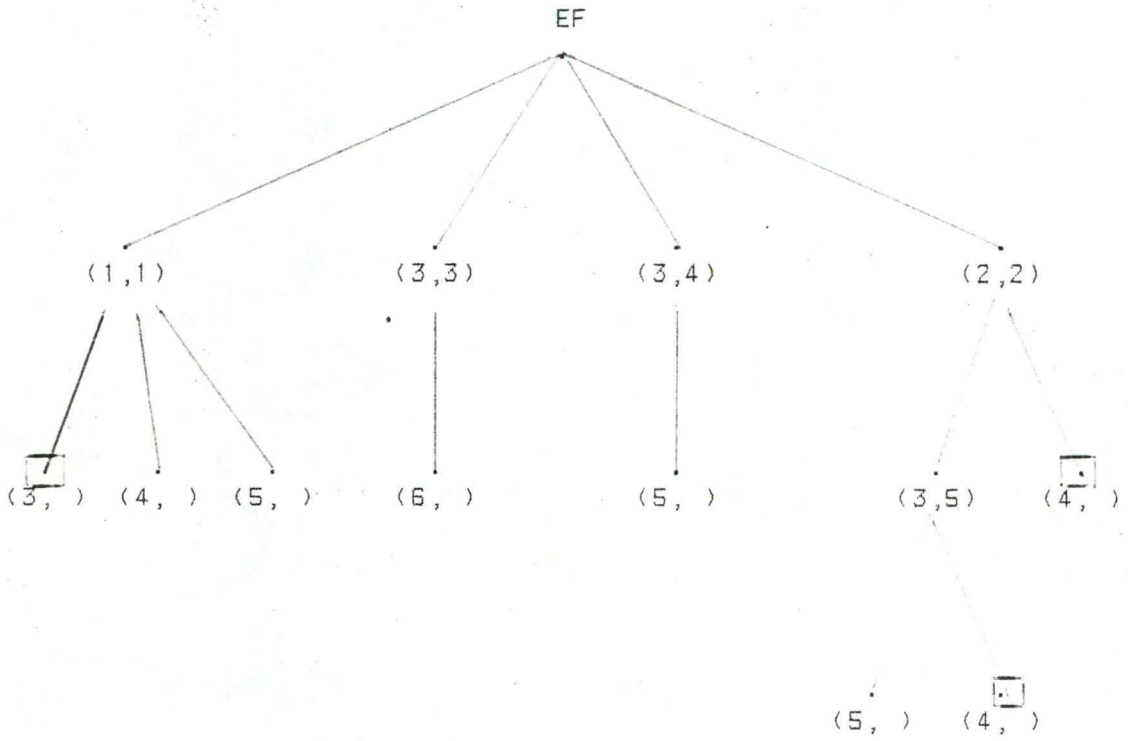
A chaque itération, l'algorithme développe un noeud, c'est-à-dire qu'il ajoute à une solution partielle, un traitement satisfaisant au moins une caractéristique technique de l'état du support. L'arbre de recherche a donc une profondeur maximum n si n est le nombre de critères techniques du support à satisfaire. Ce nombre n est fini. D'autre part, chaque noeud peut avoir au maximum P successeurs, où P est le nombre total de traitements de la gamme Trimetal. Ce nombre est également fini. L'arbre de recherche est donc de dimensions finies. La recherche parcourant au maximum tous les noeuds de l'arbre, le nombre d'itérations est fini puisqu'un noeud ne peut avoir son père comme successeur, une solution étant composée de traitements distincts.

Quel que soit le problème de peinture posé, l'algorithme se termine après un nombre fini d'itérations (proposition 3).

Exemple

Etant donné EF, l'état final du support, voyons comment la recherche se fait dans un graphe où

- tout noeud est associé à une solution candidate
- tout arc représente un traitement ajouté à la solution candidate associée au noeud dont est issu cet arc
- tout noeud est accompagné d'un couple (E,O) avec
  E, la valeur de la fonction d'évaluation de la solution candidate associée à ce noeud
  et O, le numéro d'ordre dans lequel ce noeud a été développé
- tout noeud encadré est une solution terminale.



A ce stade de la recherche, 3 solutions terminales sont obtenues et il n'existe pas de solution partielle qui soit plus intéressante puisque leur évaluation respective est supérieure à l'évaluation maximale (4) des évalutions des solutions terminales.

Si le nombre n de solutions désirées par l'utilisateur est strictement supérieur à 3, la recherche doit se poursuivre en

continuant de construire la solution partielle la plus intéressante (celle dont la valeur de la fonction d'évaluation est la plus petite).

Si le nombre n est égal à 3, les solutions terminales sont les systèmes-solution.

Si le nombre n est strictement inférieur à 3, des questions sur les critères extra-techniques avec question doivent être posées à l'utilisateur afin de choisir les n meilleures solutions parmi les solutions. Ces n systèmes de peinture sont les systèmes-solution.

Seule l'évaluation permet d'élaguer des sous-arbres. En effet, les noeuds que l'on refuse de développer sont ceux dont l'évaluation est trop grande par rapport à celles d'autres noeuds.

Comme nous le montre bien cet exemple, le système procède à l'étude simultanée de plusieurs solutions candidates. Le choix d'en poursuivre la construction est guidée par les critères économiques, matérialisés par une fonction d'évaluation.

Nous allons maintenant voir plus en détails comment s'effectue l'évaluation et comment sont déterminées les questions à poser concernant les critères extra-techniques.

IV.4.2.2.1. La fonction d'évaluation

L'objectif de ce point est de construire la fonction d'évaluation de la recherche d'un système de peinture.

Rappelons que l'évaluation est calculée après avoir pris en considération tous les critères techniques, puisque seules les solutions candidates sont "évaluées".

Dans un premier temps, nous allons voir quelles sont les informations qui peuvent être déduites de la solution partielle ou terminale à évaluer et qui doivent former la base de l'évaluation.

Un algorithme de recherche, pour être idéal, doit être admissible et optimal. Nous examinons donc ensuite comment ces exigences peuvent se traduire à travers la fonction d'évaluation. Un exemple permet d'illustrer comment ces deux impératifs sont pris en considération.

Dans un troisième temps, nous étudions comment il est possible d'accorder des importances différentes aux critères extra-techniques et de mettre leurs valeurs sur une même échelle.

Il s'agit donc d'affiner progressivement la fonction d'évaluation en tenant compte petit à petit de contraintes supplémentaires (et nécessaires). En dernier lieu, nous exposons la formule finale permettant d'évaluer la solution candidate en fonction de toutes ces contraintes.

A. ALTERNATIVES POUR LA DEFINITION D'UNE FONCTION D'EVALUATION

Plusieurs optiques sont possibles.

1. La suppression de sous-arbres se fait grâce à la fonction d'évaluation basée sur la stratégie suivante : dès qu'une solution terminale à X produits a été découverte, les solutions terminales à plus de X + 1 produits sont évaluées de manière à ce que ces solutions soient rejetées.

   EXEMPLE.

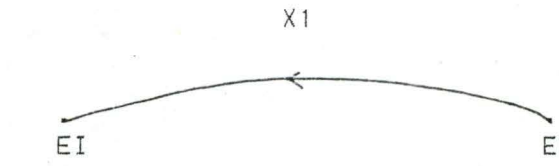
   Supposons que l'utilisateur veut peindre un support actuellement à l'état nu, et qu'il désire lui donner un aspect brillant et structuré et une résistance à l'environnement urbain et à l'abrasion forte.

   Soit EF, l'état final du support et EI, l'état initial du support.

   Supposons le nombre de solutions désirées par l'utilisateur égal à 1.

   Soient 2 solutions candidates :

   I.

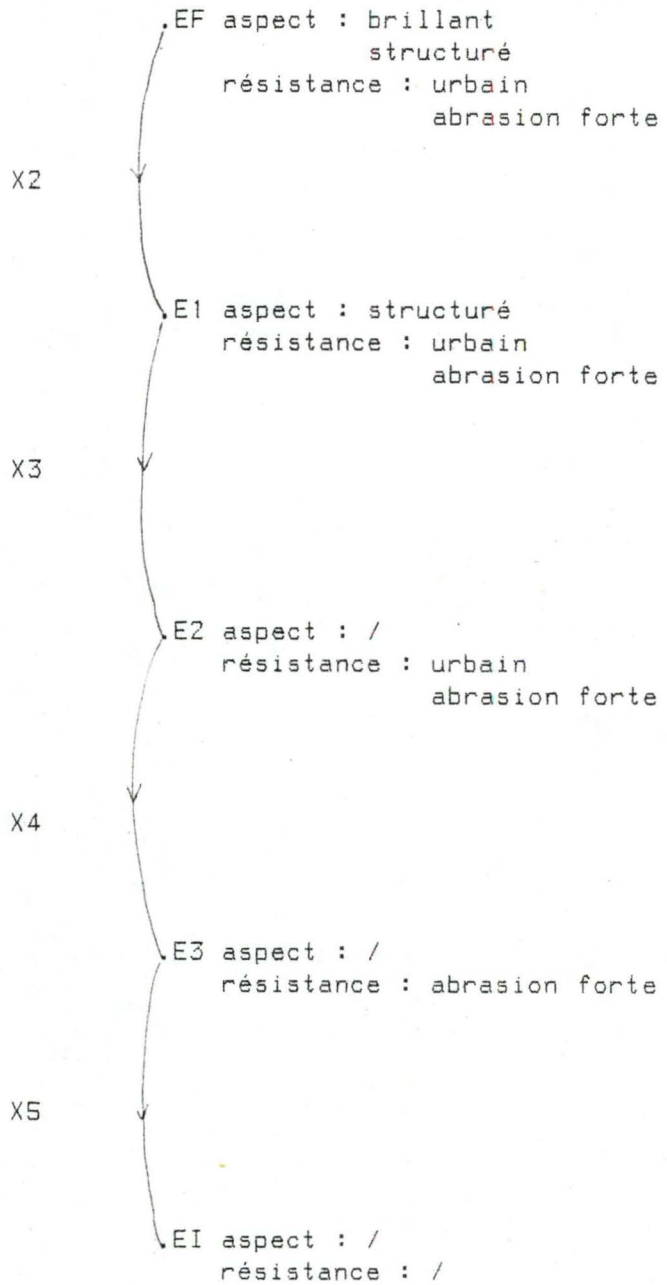


aspect : /
résistance : /

aspect : brillant
structuré
résistance : urbain
abrasion
forte

avec X1, un produit dont la post-condition est :
aspect : brillant
structuré
résistance : urbain
abrasion forte

II.



avec X2, un produit dont la post-condition est :
aspect : brillant
résistance : rural

X3, un produit dont la post-condition est :
aspect : structuré
résistance : /

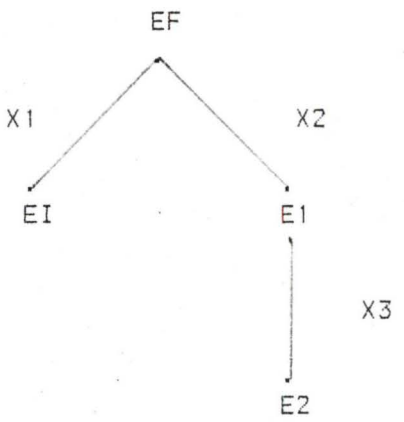
X4, un produit dont la post-condition est :
aspect : opaque
résistance : urbain

X5, un produit dont la post-condition est :

aspect : /
résistance : abrasion forte.

Cette stratégie suppose que l'utilisateur préfèrera sans aucun doute la solution I car elle comporte 2 produits en moins et donc qu'elle nécessite moins de travail pour le client.

Avec une telle fonction d'évaluation, la recherche se ferait comme suit :



- les solutions (X1) et (X2) sont générées et évaluées
- la solution (X2) est partielle, elle peut donc être développée (si son évaluation est strictement inférieure à celle de la solution (X1)). La solution partielle (X2,X3) est générée et évaluée à une valeur nettement supérieure à l'évaluation de (X1) de manière à la rendre inadmissible par rapport aux autres solutions éventuelles, puisqu'elle comportera au moins deux produits en plus que (X1), (X2, X3) n'étant pas une solution complète.

L'inconvénient de cette démarche est que le critère "simplicité du système" - nombre minimum de produits différents - est en somme le critère le plus important et est fixé comme tel une fois pour toutes. Même si actuellement ce critère est en effet le plus important, nous ne pouvons admettre cette solution car un critère y est privilégié, ce qui empêche le système d'être dynamique et adaptable. L'optique adoptée doit impérativement permettre une modification aisée de l'ordre d'importance des critères, cet ordre variant en fonction de nombreux facteurs.

Les différentes possibilités suivantes utilisent une fonction d'évaluation numérique avec notamment des coefficients d'importance, qui permettent de tenir compte d'un ordre d'importance entre les

différents critères extra-techniques sans question. Ce qui les différencie, c'est l'étendue de la fonction d'évaluation.

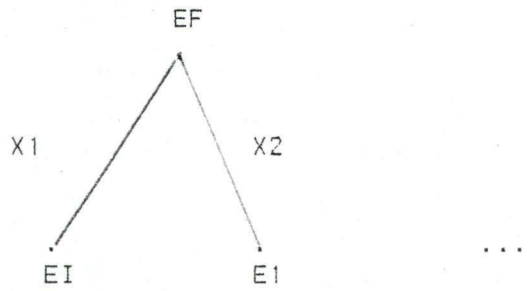
2. Etant donné une solution candidate, son évaluation se fait à partir des données explicites contenues dans cette solution.
   Définition : les données explicites d'une solution candidate sont les valeurs des critères extra-techniques sans question des produits CONSTITUANT cette solution.

   Les principaux avantages d'une telle approche sont la simplicité de l'évaluation et la certitude d'obtenir la meilleure solution.

   Mais un inconvénient majeur se pose. Toutes les connaissances dont nous pouvons disposer ne sont pas exploitées, notamment l'état courant de la recherche : ce qui a déjà été fait (les caractéristiques déjà satisfaites) et ce qu'il reste à faire (les caractéristiques non encore satisfaites), c'est-à-dire la distance qu'il reste à parcourir jusqu'à l'état initial du support. Or, afin d'explorer le moins possible de solutions candidates inintéressantes, il est nécessaire de subodorer au maximum (mais avec un minimum de certitude) quelles seront les caractéristiques des produits qui seront appliqués par après.

Exemple.

Reprenons l'exemple précédent.
Après la 1ère étape, le graphe suivant est obtenu :



L'évaluation de (X1) se fait à partir des critères extra-techniques sans question de X1, c'est-à-dire son prix, ... et uniquement à partir de ces caractéristiques. (X1) est une solution terminale.

L'évaluation de (X2) se fait de la même manière, par rapport aux caractéristiques extra-techniques de X2. (X2) est une solution partielle.

Or, le prix de X2 qui est beaucoup moins performant, est probablement nettement moins élevé que celui de X1 et de ce fait, la solution (X2) est favorisée grâce à une évaluation plus petite. Ceci n'est pas réaliste puisque la solution terminale générée à partir de X2 comportera d'autres produits pour satisfaire les éléments restants, à savoir "aspect structuré" et "résistances abrasion forte et urbain". Son évaluation s'en verra donc augmentée considérablement.

Il serait alors plus correct de tenir conpte dès la 1ère étape, des éléments qu'il reste à satisfaire pour obtenir l'état initial du support, c'est-à-dire de la "différence" entre l'état courant E1 de la solution partielle (X2) et l'état initial EI.

3. L'évaluation d'une solution candidate se base sur les données explicites et sur toutes les informations que l'on peut déduire en ce qui concerne la suite de produits à appliquer jusqu'au moment où l'état initial du support est satisfait.

Une technique pour découvrir ces informations serait de faire la "différence" entre l'état courant de la solution candidate et l'état initial afin de déterminer le nombre et la nature des contraintes techniques qu'il faut encore satisfaire. Il faudrait ensuite calculer le nombre et la nature des produits nécessaires à la réalisation complète du problème.

Exemple.

Supposons une solution candidate dont l'état courant est déterminé par l'aspect lisse, les résistances à l'environnement urbain et à l'abrasion forte et la protection contre le bleuissement. L'état initial du support est caractérisé par une protection contre l'environnement urbain. La "différence" entre ces deux états, représentant les contraintes qu'il reste à satisfaire sont : aspect lisse, résistances à l'environnement urbain et à l'abrasion forte. Selon cette stratégie, il faudrait, en consultant la base des produits, si ces caractéristiques peuvent être satisfaites par un seul produit et quelles seraient les caractéristiques extra-techniques de celui-ci. Si un seul produit n'est pas suffisant, il faudrait déterminer quelles sont les caractéristiques extra-techniques des premiers produits à sélectionner et celles des produits qui pourraient satisfaire les contraintes restantes.

L'avantage de cette méthode est la réduction importante de noeuds à visiter. Cette réduction a malheureusement l'inconvénient suivant : il est quasi impossible de déterminer avec une certitude satisfaisante quels seront les caractéristiques de tous les produits à appliquer après l'état courant de la solution candidate à évaluer. En supposant même qu'une telle chose soit possible, le gain de temps provoqué par la réduction de l'effort de recherche serait perdu par le temps de calcul de la valeur de la fonction

d'évaluation d'un état courant.

4. La stratégie que nous avons adoptée est un compromis entre ces deux dernières positions. L'évaluation tient compte d'une part des informations explicites contenues dans la solution candidate dont il est question, et d'autre part des caractéristiques qui devront tôt ou tard être satisfaites par les prochains produits candidats, si l'ajout d'un produit est nécessaire. En effet, si la solution candidate satisfait tous les composants de l'état initial, le processus de recherche est terminé pour ce chemin et la solution associée à ce dernier peut être évaluée à sa juste valeur. Dans le cas contraire, on est certain qu'un (que des) produit(s) supplémentaire(s) est (sont) nécessaire(s). Dans ce cas, le prochain produit sélectionné pour amener un élément à la solution, doit satisfaire au moins un critère technique de l'état final non encore satisfait par la solution candidate. Il est alors possible de déduire certaines informations (ces informations seront déterminées avec précision dans les points suivants) concernant les critères extra-techniques sans question de ce produit et de les introduire dans l'évaluation. Dès lors, à l'évaluation explicite de la solution candidate, peut être ajoutée une évaluation des caractéristiques du prochain traitement à appliquer. Ces caractéristiques sont appelées données implicites. Il s'agit de la deuxième alternative complétée de données supplémentaires : les données implicites; la valeur de la fonction d'évaluation pour une solution est une agrégation des données explicites et implicites. Le coût d'un traitement est une agrégation des valeurs de ses critères extra-techniques sans question et le coût d'une solution est la somme des coûts des traitements qui la composent.

   Exemple : dans l'exemple repris à la section IV.4.2.2.1, l'évaluation de la solution partielle (X2) doit contenir :

   - les données explicites : le prix de X2, son temps de séchage, ...
   - les données implicites : le fait que le prochain produit devra
     soit avoir un aspect structuré
     soit procurer une résistance à l'abrasion forte
     soit procurer une résistance à l'environnement urbain.

Ces données implicites permettent de limiter le nombre de produits candidats pour l'étape suivante de la recherche et donc de mieux cerner ce prochain produit et ses caractéristiques.

Forme générale de la fonction d'évaluation

La valeur totale de la fonction d'évaluation d'une solution candidate est la somme des évaluations pour chaque critère extra-technique sans question.

L'évaluation d'un critère extra-technique sans question pour une solution candidate est la somme d'une valeur explicite et d'une valeur implicite.

1) Valeur explicite (évaluation des données explicites)
La solution candidate est constituée des traitements déjà calculés pour chacun desquels nous pouvons trouver les valeurs des critères extra-techniques sans question dans leur vecteur d'effets de bord. La valeur explicite est tout simplement la somme des valeurs de ce critère pour chaque produit de la solution.

2) Valeur implicite (évaluation des données implicites)
Le calcul de la valeur implicite n'a lieu que si la solution candidate à évaluer n'est pas terminale. Cette valeur implicite est l'évaluation de la valeur de ce critère pour le prochain traitement qui sera sélectionné. Elle sera déterminée au cours des points B et C.

L'évaluation d'un critère s'obtient

* en additionnant la valeur explicite et la valeur implicite
* et en multipliant cette somme par les coefficients de pondération et d'importance correspondant à ce critère. Nous verrons pourquoi de tels coefficients sont nécessaires et comment ils peuvent être obtenus aux points E et F.

Soit une solution candidate Sc contenant m produits.

	Evaluation des données explicites	Evaluation des données implicites	Coefficient de pondération	Coefficient d'importance
- critère 1 --->	vex_1	vim_1	p_1	c_1
- critère 2 --->	vex_2	vim_2	p_2	c_2
...				
- critère n --->	vex_n	vim_n	p_n	c_n

avec * critère i, un critère extra-technique sans question,

$$* \; vex_i = \sum_{j=1}^{m} Val_{ij}$$

où Val_{ij} est la valeur du critère i pour le produit j

* vim , la valeur implicite du critère i pour le
  i prochain traitement.

La valeur de la fonction d'évaluation pour la solution courante est donc :

$$E_1 = \sum_{i=1}^{n} (vex_i + vim_i) * p_i * c_i$$

avec i, l'indice d'un critère extra-technique sans question.

Les éléments de cette formule sont affinés au cours des points suivants. L'indice de E détermine le numéro de la version.

Si la solution à évaluer est terminale, il n'y a pas de valeur implicite à calculer puisque une telle solution ne nécessite plus de traitement.

Son évaluation est donc obtenue par

$$E'_1 = \sum_{i=1}^{n} vex_i * p_i * c_i$$

Toute recherche et donc, toute fonction d'évaluation intervenant dans cette recherche, doit avoir certaines qualités dont il faut tenir compte lors de sa construction. Ces qualités vont permettre de déterminer exactement les données implicites et donc la valeur exacte de la fonctin d'évaluation pour une solution. C'est ce que nous allons examiner dans les points suivants.

B. ADMISSIBILITE DE L'ALGORITHME

Dans l'intérêt de la firme, il est capital que le programme donne toujours la meilleure solution (celle pour laquelle la valeur de la fonction d'évaluation est minimale). Il ne peut pas se contenter d'une solution qui est satisfaisante mais qui n'est pas la meilleure.

Comme nous l'avons vu au point IV.3.3.A., si une méthode heuristique n'est pas construite avec le plus grand soin, elle risque de ne pas trouver la meilleure solution. Afin de ne pas tomber dans ce piège, et donc d'assurer la propriété d'admissibilité, nous faisons appel au théorème classique suivant [NIL71] :

"Dans la mesure où il existe au moins un chemin s0 --> g et où tout arc dans le graphe d'espace d'états a un coût unitaire > 0, alors le programme de recherche trouve un chemin s0 -> g de coût minimal si pour tout s :

$$h'(s) <= h(s)$$

où h (s) est le coût réel d'un chemin de s vers g de coût minimal
et
où h' (s) est l'évaluation du coût d'un chemin de s vers g de coût minimal".

Ce théorème peut se traduire de la façon suivante : pour qu'un programme soit admissible, l'évaluation de tout noeud doit se faire de manière à ce que le coût du chemin qu'il reste à parcourir à partir de ce noeud, soit sous-évalué par rapport à son coût réel.

Si la solution candidate n'est pas une solution terminale, nous avons la certitude qu'un produit supplémentaire est nécessaire mais pour mieux évaluer cette solution candidate, nous devons déduire des informations sur les critères extra-techniques sans question de ce traitement supplémentaire. En se basant sur le théorème ci-dessus, nous pouvons affirmer que si pour chaque critère extra-technique de ce traitement, on associe la valeur minimum par rapport à TOUS les traitements de la base de données, le programme est certainement admissible.

La valeur minimum d'un critère extra-technique correspond à la valeur la plus favorable au sens défini au point IV.2.2.2. Pour les critères extra-techniques sans question, il s'agit du sens le plus naturel : prix le plus faible, ... et pour les critères extra-techniques avec question, il s'agit du sens correspondant aux préférences de l'utilisateur. Pour connaître celles-ci, il faudra lui poser quelques questions (cfr IV.4.2.2.2). Les valeurs défavorables sont celles qui vont dans le sens inverse. Pour les critères extra-techniques sans question, il s'agit du prix le plus élevé, ... et pour les critères extra-techniques avec question, il s'agit du sens ne correspondant pas aux préférences de l'utilisateur.
Les valeurs minimales à la base de l'évaluation seront affinées au point C. ci-dessous.

Reprenons les critères extra-techniques sans question et leurs valeurs minimales qui permettent d'effectuer l'évaluation des données implicites.

- Pour le prix/m2, la valeur minimale est le prix/m2 du traitement le moins cher de la base de données.

- Pour la simplicité du système, on se limite à savoir si un produit supplémentaire est nécessaire ou non. Puisque nous pouvons répondre à cette question avec certitude, l'évaluation de la promesse d'un noeud, en ce qui concerne ce critère, est la promesse réelle.

- Pour la composition du produit, la valeur minimale est celle correspondant au fait que le prochain produit est monocomposant. Cette évaluation est très proche de la réalité car il existe très peu de produits à plusieurs composants parmis les produits standards.

- Pour les promotions, la valeur minimale est celle correspondant au fait que le prochain produit est en promotion.

- Pour la base du produit, la valeur minimale est celle correspondant au fait que le prochain est de même base que la solution courante.

- Pour le temps de séchage, la valeur minimale est le temps de séchage le plus faible de tous les traitements de la base de données.

- Pour la facilité d'application, la valeur minimale est celle correspondant au fait que le produit est très aisé à appliquer.

La forme générale de la fonction d'évaluation devient

$$E_2 = \sum_{i=1}^{n} \left(vex_i + \min_{\text{pour tous les produits de la base de données}} Val_{ij} \right) * p_i * c_j$$

avec Val_{ij}, la valeur du critère i extra-technique sans question du produit j.

Montrons à l'aide du théorème de l'admissibilité, que la recherche R2 se basant sur E_2 est admissible.

Supposons une solution candidate S dont le dernier traitement sélectionné est T et pour laquelle il reste N contraintes techniques à satisfaire. Cette solution doit donc être complétée par au moins un traitement.

$$h(S) = \sum_i \sum_p Val_{ip} * p_i * c_i$$

où p est l'indice d'un traitement encore à sélectionner, en sachant qu'il en reste au minimum un pour satisfaire N contraintes.

Or $Val_{ip} <= \min_j Val_{ij}$ (j est l'indice d'un produit de la base de données)

et donc $h(S) >= h'_{R2}(S) = \sum_i (\min_j Val_{ij}) * p_i * c_i$.

L'algorithme de recherche est donc admissible.

C. OPTIMALITE DE L'ALGORITHME

Toute recherche heuristique, pour être idéale, doit non seulement être admissible, mais également optimale de manière à atteindre le plus rapidement possible la meilleure solution en développant un minimum de noeuds. Afin d'assurer la propriété d'optimalité, nous faisons appel à une définition et à un théorème classique de [NIL71].

Définition :

L'algorithme B est plus informé que l'algorithme A
ssi $h'_A(s) <= h'_B(s)$, pour tout s

où s est un noeud,
$h'_A(s)$ est l'évaluation du coût d'un chemin de s vers g (noeud but) de coût minimal, ce chemin ayant été déterminé par l'algorithme A,
$h'_B(s)$ est l'évaluation du coût d'un chemin de s vers g de coût minimal, ce chemin ayant été déterminé par l'algorithme B.

Théorème :

"Si - (hypothèse 1) A et B, deux algorithmes admissibles tels que B est plus informé que A,
- (hypothèse 2) pour tout s, s', il existe un chemin de s à s' tel que
$h'_B(s) <= (h'_B(s') +$ coût effectif d'un chemin de s à s')
alors tout noeud développé par B est aussi développé par A."

Si les conditions de ce théorème sont remplies, B développe moins de noeuds que A.

Une recherche est d'autant plus optimale qu'elle est informée. Il est donc important pour notre algorithme de connaître à chaque étape de la recherche, un maximum d'informations sur la solution qui est en train d'être construite et ainsi de savoir si elle est préférable à toute autre solution en cours.

Or, si pour chaque solution non terminale, on affecte toujours la valeur minimale (par rapport à l'ensemble des produits de la base de

données) pour chaque critère, seule une distinction entre les solutions partielles et les solutions terminales a lieu. Aucune solution partielle n'est revalorisée ou dépréciée par rapport aux autres.

Afin d'évaluer au mieux chaque solution candidate, nous allons restreindre l'ensemble des produits susceptibles d'être candidats pour la prochaine sélection d'un traitement, grâce aux contraintes non encore satisfaites par la solution. Le minimum des valeurs des critères sur lesquelles se base l'évaluation sera calculé sur cet ensemble, au lieu d'être calculé sur l'ensemble de la base de données.

L'élaboration de cet ensemble pour chaque solution candidate est construit à l'aide de classes. Nous allons tenter de définir la notion de classe qui détermine l'ensemble des produits susceptibles d'être candidats pour compléter la solution qui est en train d'être étudiée.

C1. La notion de classe

Etant donné les contraintes qu'il reste à satisfaire (ces contraintes constituent la "différence" entre l'état courant de la solution candidate et l'état initial), nous pouvons affirmer que le prochain produit sélectionné accomplira une de ces contraintes et que tôt ou tard toutes ces contraintes seront satisfaites.

Ceci mène à une première approximation de la notion de classe : une classe est une collection de produits qui satisfont un ensemble de critères techniques. Un même ensemble de critères ne peut être associé à plusieurs classes.

Selon le nombre de critères techniques, le contenu de la classe correspondante diffère. En effet, pour une classe, plus ce nombre est élevé, plus les produits doivent satisfaire d'exigences et donc, moins grand est le nombre de tels produits. La taille de l'ensemble des critères détermine également le nombre de classes à envisager : si N critères sont admis par classe et si le nombre total de critères est M (M>=N), il doit exister autant de classes que de combinaisons de N critères parmi M. Il ne serait pas logique de ne pas considérer toutes les combinaisons possibles puisqu'elles peuvent toutes se produire lorsqu'on considère les contraintes qu'il reste à satisfaire pour une solution particulière.

Exemple.

S'il existe trois critères techniques
- aspect brillant,
- protection contre la corrosion,
- résistance au lavage moyen,

et si le nombre de critères associés à une classe est deux, alors

les classes à envisager sont les suivantes :

- classe de produits dont l'aspect est brillant et qui assurent une protection contre la corrosion
- classe de produits dont l'aspect est brillant et qui résistent au lavage moyen
- classe de produits qui assurent une protection contre la corrosion et qui résistent au lavage moyen.

Afin de définir de manière unique une classe, il convient de fixer le nombre de critères associés à une classe. Remarquons que plus le nombre de critères est élevé,

* plus le nombre de classes est élevé et
* plus le nombre de produits par classe est restreint et donc, plus le prochain produit susceptible de convenir est déterminé de manière précise.

Le choix du nombre de critères par classe et donc du nombre de classes, dépend de la précision que nous voulons octroyer à la fonction d'évaluation.

Plusieurs optiques dont deux extrêmes sont envisageables.

* Si nous prenons une seule classe comprenant tous les produits, pour chaque état courant, la partie de l'évaluation basée sur les données implicites sera la même pour toute solution candidate, puisque les valeurs sur lesquelles elle se base sont uniques. De ce fait, les états courants ne se différencieraient que par les données explicites, ce qui est une stratégie que nous nous étions refusée.

* Par contre, si nous avons une classe définie pour chaque combinaison possible de tous les critères techniques, le nombre de classes est nettement plus élevé. Par conséquent l'évaluation des solutions sera beaucoup plus précise. En effet, à une solution correspond une seule classe et l'évaluation de chaque solution candidate se différencie donc des autres. Le nombre de classes serait énorme. Le temps de recherche de la bonne classe serait trop long et inadmissible pour une fonction d'évaluation.

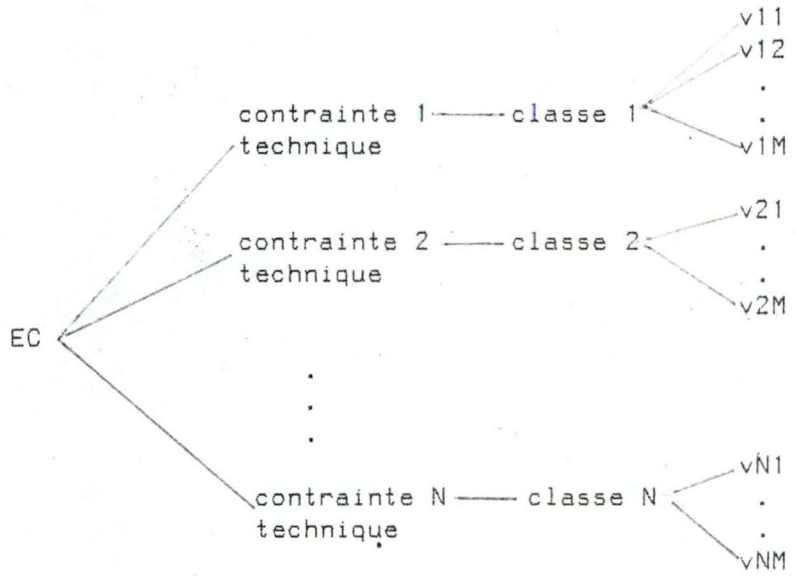
* Le compromis entre ces deux solutions est de définir une classe par contrainte. Ainsi, toute solution candidate serait évaluée au moyen d'un ensemble de classes : ces classes représentent les caractéristiques de l'état courant non encore satisfaites, et/ou les caractéristiques "permanentes" de l'état courant. Nous avons choisi cette solution car elle est plus précise que la première (on ne prend pas tous les produits) et plus rapide que la deuxième.

Ceci nous mène à la définition d'une classe : une classe est un ensemble de produits qui satisfont un même critère technique. Un critère ne peut être associé à plusieurs classes.

Par définition, un produit qui satisfait i contraintes techniques distinctes, appartient à i classes.

Chaque contrainte technique détermine une classe de produits qui respectent la contrainte considérée. Pour procéder à l'évaluation, à chaque classe sont associés les minimums des valeurs des critères extra-techniques sans question pour les traitements contenus dans cette classe.

Les classes existantes et leur valeur minimale par rapport aux critères extra-techniques sans question peuvent être schématisées de la manière suivante, en supposant que N est le nombre de contraintes techniques qu'il reste satisfaire pour l'état EC et M est le nombre de critères extra-techniques sans question.



où EC est l'état courant d'une solution candidate,
contrainte k est un critère technique qu'il faut satisfaire,
classe k, la classe des produits satisfaisant la contrainte technique k,
vki est le minimum des valeurs du critère extra-technique i (sans question) pour les produits de la classe k.

Exemple : soit la contrainte k, la protection contre la corrosion. La classe k est donc l'ensemble des produits qui fournissent une protection contre la corrosion.

vk1 est le prix minimum parmi les produits de cette classe et vk5 est le temps de séchage minimum parmi ces produits.

Ce qu'il est important de connaître pour chaque classe n'est pas l'ensemble des produits qu'elle représente, mais bien les valeurs minimums de leurs caractéristiques extra-techniques. C'est la raison pour laquelle la classe associée au critère technique CT se formalise de la manière suivante, si CETi est le nom du ième critère extra-

technique sans question associé à la classe et si VALi est la valeur minimum de ce ième critère :

classe (CT, [[CET1, VAL1], ... , [[CETi, VALi], ... [CETn, VALn]]).

avec CT, le nom du critère technique associé à la classe,
CETi, le nom du ième critère extra-technique sans question,
VALi, la valeur minimum du ième critère extra-technique sans question.

Exemple.

La classe associée au critère technique "protection contre la corrosion" est représentée par le fait suivant :

classe (corrosion, [[pm, 26], [tsech, 1], [prom, 0], [comp, mono], [fac, oui]]).

avec pm signifiant prix au mètre carré,
tsech, signifiant temps de séchage,
prom, signifiant promotion,
comp, signifiant composition,
fac, signifiant facilité,
mono, signifiant mono-composant.

C2. Valeurs choisies pour la fonction d'évaluation

Etant donné les contraintes k encore à satisfaire, les classes k et les valeurs v_{ki} correspondantes, comment caractériser le prochain produit qui sera sélectionné ?

La valeur du critère i extra-technique sans question que nous prenons en considération pour l'évaluation, est le maximum pour l'ensemble des classes des valeurs de ce critère, c'est-à-dire

$$\max_{\text{classe } k} \min_{\text{produits de classe } k} Val_{ij} = \max_{\text{classe } k} v_{ki}$$

où Val_{ij} est la valeur du critère i extra-technique sans question du produit j.

Le maximum correspond en fait à la valeur la plus défavorable de ce critère. Comme cette contrainte doit être satisfaite tôt ou tard, la solution finale comprendra indiscutablement un traitement ayant la valeur de ce maximum. L'évaluation est alors assez précise tout en restant admissible.

Reprenons les critères extra-techniques et les valeurs qui permettront d'effectuer l'évaluation des données implicites.

- Pour le prix/m2, la valeur maximale (la valeur à considérer pour l'évaluation) est le maximum des prix/m2 obtenus en recherchant le minimum des valeurs de ce critère pour chaque classe.

- Pour la simplicité du système, la notion de classe n'intervient pas.

- Pour la composition du produit, la valeur maximale est celle correspondant au fait qu'il existe une classe qui ne contient que des produits à plusieurs composants. S'il n'existe pas de telle classe, la valeur à considérer est la valeur associée au fait qu'un des prochains produits sélectionnés est un mono-composant (afin de garder un programme admissible).

- Pour les promotions, la valeur maximale est celle correspondant au fait qu'il existe une classe qui ne contient que des produits qui ne sont pas en promotion. S'il n'existe pas de telle classe, la valeur considérée est la valeur associée au fait qu'un des prochains produits sélectionnés est un produit en promotion (afin de garder un programme admissible).

- Pour la base du produit, il existe un nombre équivalent de traitements à base solvants et à base acqueuse. De sorte que chaque classe contient autant de traitements de chaque sorte. Dès lors, nous ne pouvons rien déduire en ce qui concerne la base des produits qui seront sélectionnés.

  La base n'intervient dans l'évaluation qu'au niveau des données explicites. A ce moment, nous évaluons à sa juste valeur le fait que la solution comprend des produits de bases différentes ou de même base (ce dernier cas étant plus favorable).

- Pour le temps de séchage, la valeur maximale est le maximum des temps de séchage obtenus en recherchant le minimum des valeurs de ce critère pour chaque classe.

- Pour la facilité d'application, la valeur maximale est celle correspondant au fait qu'il existe une classe qui ne contient que des produits qui sont difficiles à appliquer. S'il n'existe pas de telle classe, la valeur considérée est la valeur associée au fait qu'un des prochains produits sélectionnés est un produit aisé à appliquer (afin de garder un programme admissible).

La forme générale de l'évaluation devient

$$E_3 = \sum_{i=1}^{n} (vex_i + max\ v_{ki}) * p_i * c_i$$

Nous allons montrer à l'aide du théorème de l'optimalité que la recherche R3 se basant sur E_3 développe moins de noeuds que la recherche R2 qui se base sur l'évaluation E_2 définie au point B..

Cette affirmation est démontrée si les hypothèses du théorème sont satisfaites.

- hypothèse 1

R2 est admissible.

Montrons que R3 est admissible.
Soit S, une solution partielle.

$$\sum_i (\max_{ki} v_{ki}) * p_i * c_i = h'_{R3}(S) <= h(S)$$

puisque $\max_{ki} v_{ki} = v_{ji}$ est la valeur minimum d'une classe j. Un produit de cette classe devra nécessairement être sélectionné pour satisfaire la contrainte j.
Or, $v_{ji} = \min Val_{ip}$ où p est l'indice d'un produit de la classe j et donc v_{ji} est inférieur ou égal au Val du produit qui sera réellement sélectionné.

Donc, R3 est admissible.

R3 est plus informé que R2 puisque $h'_{R3}(s) >= h'_{R2}(s)$.

En effet, $h'_{R3}(s) = \sum_i (\max_{ki} v_{ki}) * p_i * c_i$

$$h'_{R2}(s) = \sum_i \left(\min_{\substack{\text{pour tous} \\ \text{les produits} \\ \text{de la base de} \\ \text{données}}} Val_{ij} \right) * p_i * c_i$$

et pour tout i, on a

$$\max_k v_{ki} = \left(\max_k \min_{\text{produits de la classe } k} Val_{ij}\right) >= \left(\min_{\substack{\text{pour tous} \\ \text{les produits} \\ \text{de la base de} \\ \text{données}}} Val_{ij} \right)$$

Cette dernière inégalité est correcte puisque la classe k est un sous-ensemble de la base de données.

- hypothèse 2

Soit une solution candidate S' comprenant deux traitements T et T', et avec T sélectionné avant T'.
Soit une solution candidate S comprise dans S' et se terminant à T.
Supposons également que le nombre de contraintes techniques qu'il reste à satisfaire après sélection de T est égal à N. Dès lors, le nombre de contraintes techniques qu'il reste à satisfaire après sélection de T' est au plus égal à N-1, T' satisfaisant au moins une contrainte (dans le cas contraire , il n'aurait pas été sélectionné).

Nous pouvons alors affirmer que

$$h'_{R3}(T') + cm(T,T') \geq h'_{R3}(T)$$

En effet, $h'_{R3}(T) = \sum_{i=1}^{n} (\max v_k) * p_{ki} * c_i$

où k est l'indice de la classe correspondant au critère technique k.

$$h'_{R3}(T') = \sum_{i=1}^{n} (\max v_k) * p_{ki} * c_i$$

où k est l'indice de la classe correspondant au critère technique k.

Donc, $h'_{R3}(T) \geq h'_{R3}(T')$

puisque pour T' le $\max v_k$ est calculé sur un sous-ensemble des classes intervenant pour T.

cm(T,T') représente la somme des critères extra-techniques du traitement T' et éventuellement de ceux qui ont été sélectionnés entre T et T'. Cette somme ne fait que renforcer l'inégalité démontrée.

L'algorithme de recherche se basant sur E_2 développe au moins autant de noeuds que l'algorithme de recherche se basant sur E_3.

C3. Constitution des classes

Le contenu des classes étant constant quel que soit le moment de la recherche, ces dernières peuvent être constituées lors de l'initialisation du système. Leurs minima peuvent donc être stockés en mémoire afin d'éviter toute perte de temps provoquée par l'exécution, à plusieurs reprises, de calculs identiques. Les minima seront ajustés lors de l'ajout ou du retrait d'un traitement quelconque.

Les minima pourraient être ordonnés par critère et par ordre croissant, ce qui faciliterait la recherche du maximum lors de l'évaluation implicite.

D. EXEMPLE

Soit une classe dont le critère technique est "lavage moyen". Cette classe contient 6 produits différents.

produit	temps de séchage	composition	facilité	prix
Stellapaint	15 h	M	F	31.76 fr/m2
Tpcextrapale	15 h	M	D	32.36 fr/m2
Tpcmat	15 h	M	F	28.43 fr/m2
Tpcsatin	15 h	M	F	29.58 fr/m2
Tpcyacht	15 h	M	D	33.78 fr/m2
Xyladécor	18 h	M	F	27.55 fr/m2

La classe "lavage moyen" aurait comme minima :

temps de séchage -------> 15 h
composante -------------> M
facilité d'application -> F
prix -------------------> 27.55 fr/m2

Soient d'autres classes et leurs minima :

classe	temps de séchage minimum	composition	facilité	prix
bleuissement	13 h	M	D	35 fr/m2
primaire	6 h	M	F	25 fr/m2
Ultra violet	28 h	M	F	29 fr/m2
lavage moyen	15 h	M	F	27 fr/m2
acide	/	/	/	/

Voici un exemple de deux cas qui peuvent se présenter.

A) L'utilisateur demande une protection contre le bleuissement, une résistance au lavage moyen et un aspect brillant.

Le système, après avoir choisi le produit à mettre en surface calcule la valeur de la fonction d'évaluation pour les états courants de différentes solutions. Soient 3 états courants correspondant à 3 solutions partielles : Ec1, Ec2, Ec3.

Pour Ec1, il reste à satisfaire les critères techniques : bleuissement, primaire, lavage moyen.
Pour Ec2, il reste à satisfaire le critère technique : bleuissement.
Pour Ec3, il reste à satisfaire les critères techniques : primaire, lavage moyen.

* Pour l'état courant EC1, trois critères techniques n'ont pas encore été satisfaits.

Si nous prenons les valeurs maxima de chacune des classes (bleuissement, primaire, lavage moyen) et uniquement celles-là puisque seules ces contraintes restent à satisfaire, nous avons pour chaque critère extra-technique :

35 fr/m2, le prix minimum d'un produit de la classe "bleuissement",

15 h, le temps de séchage minimum de la classe "lavage moyen",

D, la facilité d'application de tous les produits de la classe "bleuissement",

M, un produit monocomposant.

Entre l'état courant Ec1 et l'état initial, on aura un produit dont le prix minimum sera de 35 fr/m2, le temps de séchage supérieur ou égal à 15 h. Il y aura des produits monocomposants mais on aura au moins un produit difficile à appliquer.

La valeur de la fonction d'évaluation pour Ec1 se calcule comme suit :

$$(vex_1) + (35\ fr/m2 * p_1 * c_1) + (15\ h * p_2 * c_2) + (D * p_3 * c_3) + (m * p_4 * c_4)$$

où vex est l'"évaluation" explicite de la solution courante associée à Ec1

$p_1, c_1, p_2, c_2, p_3, c_3, p_4, c_4$ représentent les coefficients de pondération et d'importance pour, respectivement, le prix au m2, le temps de séchage, la facilité d'application et la composition (cfr E. et D.).

* Pour Ec2, la valeur de la fonction d'évaluation est :

$$(vex_2) + (35\ fr/m2 * p_1 * c_1) + (15\ h * p_2 * c_2) + (D * p_3 * c_3) + (M * p_4 * c_4)$$

* pour Ec3, la valeur de la fonction d'évaluation est :

$$(vex_3) + (27\ fr/m2 * p_1 * c_1) + (15\ h * p_2 * c_2) + (F * p_3 * c_3) + (M * p_4 * c_4)$$

B) L'utilisateur demande une protection au bleuissement et à l'acide.

Le système, après avoir choisi le produit à mettre en surface, qui contient une protection contre le bleuissement, calcule la valeur de la fonction d'évaluation pour les états courants de différentes solutions.

Supposons que pour Ec1, il reste à satisfaire "acide" et que pour Ec2, il reste à satisfaire "primaire et acide".

Or, la gamme Trimetal n'offre pas de protection à l'acide. Il n'existe pas de solution pour ce problème, puisque la classe "acide" ne contient pas d'éléments. L'évaluation de ces états courants, nous donnera le même résultat : "pas de solution".

Dans la forme générale de la fonction d'évaluation, seuls les éléments p_i et c_i restent à déterminer. C' est ce que nous allons aborder dans les deux points suivants.

E. LES COEFFICIENTS D'IMPORTANCE

Les critères extra-techniques sont classés par ordre d'importance. L'ordre des critères extra-techniques avec question détermine l'ordre dans lequel il faudra poser des questions à l'utilisateur pour départager plusieurs solutions optimales. L'ordre des critères extra-techniques sans question détermine les solutions courantes à examiner en premier lieu. L'ordre de ces critères est représenté, dans la fonction d'évaluation, par des "coefficients d'importance". Ceux-ci permettent de valoriser les solutions pour lesquelles un critère important a la valeur la plus favorable, et de dévaloriser les solutions pour lesquelles un critère important a la valeur la plus défavorable.

Cet ordre (et donc la valeur des p_i) est déterminé lors d'une phase que nous appelons "initialisation du système", et peut être modifié à tout moment (mais en dehors de la résolution d'un problème) par une personne ayant le droit de modifier ces coefficients d'importance.

L'importance des critères permet en outre de faire la distinction entre les différents types de clients et lieux de vente, en fonction desquels les préférences varient.

Exemple :
- dans un supermarché, le critère le plus important est le critère "prix", celui-ci devant être minimum
- dans une droguerie, le facteur "qualité" est le plus important
- le professionnel demande en premier lieu un prix faible et en second lieu des produits anciens
- le particulier accorde énormément d'importance au rapport qualité/prix.

Cet ordre permet, de temir compte des différents souhaits et préférences des sociétés Trimetal (Trimetal Belgique, Trimetal France, Trimetal Nederland, Trimetal Italie et Mäder Bénélux).

Le gros avantage de ces coefficients est qu'ils rendent très aisée une modification de l'ordre d'importance des critères. Cet aspect est essentiel puisque cet ordre varie comme nous venons de le voir, en fonction du type de client et du lieu de vente, mais également en fonction du temps. En effet, avant la crise, les critères les plus importants étaient la qualité et l'aspect décoratif. Par contre, en période de crise, le client préfère une peinture de moins bonne qualité et moins décorative si elle est moins coûteuse. Actuellement, c'est donc le critère "prix" qui est prédominant.

Dès lors, il est indispensable de pouvoir modifier facilement les coefficients de la fonction d'évaluation afin de rendre le système adaptable à différents facteurs.

F. LES COEFFICIENTS DE PONDERATION

L'évaluation d'une solution courante ne peut être simplement la somme des évaluations obtenues pour chaque critère extra-technique. En effet, un critère qui admet des valeurs élevées, risque très fort de devenir malgré lui, le critère le plus important.

Des coefficients de pondération sont donc nécessaires afin que les valeurs de tous les critères extra-techniques sans question soient sur la même échelle.

Pour chaque critère, une unité est définie. Il s'agit alors de se choisir un critère et de ramener tous les autres critères à l'unité du critère pris comme référence.

Celui-ci doit être un critère admettant plus de 2 valeurs, dont une est la plus favorable, une autre la plus défavorable et une troisième est la valeur moyenne, c'est-à-dire celle que l'on trouve en moyenne dans un produit ou dans un système de peinture, selon la nature du critère.

Pour un critère admettant plus de 2 valeurs (tels le prix/m2, le nombre de produits et le temps de séchage) :

- nous devons connaître sa valeur minimum (valeur la plus favorable) et sa valeur moyenne

- pour connaître le coefficient de pondération relatif à ce critère, il suffit
  * de faire la différence entre la valeur moyenne et la valeur minimum du critère pris en référence.
  * de faire la différence entre la valeur moyenne et la valeur minimum du critère pour lequel on cherche le coefficient de pondération
  * enfin, de faire la division entre ces deux différences.

Pour un critère n'admettant que 2 valeurs (telles la base, la composition, la promotion et la facilité d'application) :

- nous devons connaître ses 2 valeurs c'est-à-dire, la valeur favorable et la valeur défavorable; il faut également déterminer parmi ces deux valeurs, quelle est la valeur moyenne

- pour connaître le coefficient de pondération relatif à ce critère,
  si la valeur moyenne de ce critère est la valeur la plus favorable , il suffit
  * de faire la différence entre la valeur moyenne et la valeur minimum du critère pris en référence
  * de faire la différence entre les deux valeurs du critère pour lequel on cherche le coefficient de pondération
  * ensuite de faire la division entre ces deux différences

  si la valeur moyenne de ce critère est la valeur la plus défavorable, il suffit
  * de faire la différence entre la valeur maximum et la valeur

moyenne du critère pris en référence
* de faire la différence entre les deux valeurs du critère pour lequel on cherche le coefficient de pondération
* ensuite de faire la division entre ces deux différences.

Le coefficient de pondération du critère pris en référence est bien sûr égal à 1.

Si pour le critère pris en référence, la valeur minimum est A,
la valeur moyenne est B
la valeur maximum est C,

pour un critère i admettant plus de deux valeurs,
la valeur minimum est Ai,
la valeur moyenne est Bi,

pour un critère j admettant deux valeurs Dj et Ej dont la moyenne est la plus favorable,

pour un critère k admettant deux valeurs Fk et Gk dont la moyenne est la plus défavorable,

alors le coefficient de pondération p_i du critère i est $\frac{B - A}{Bi - Ai}$

le coefficient de pondération p_j du critère j est $\frac{B - A}{Ej - Dj}$

le coefficient de pondération p_k du critère k est $\frac{C - B}{Gk - Fk}$

Exemple : soit "nombre de produits", le critère pris en référence.

Unités : - prix/m2 --> franc par m2
- temps de séchage --> heure
- nombre de produits --> nombre de produits
- composition --> néant
- promotion --> néant
- base --> néant
- facilité d'application --> néant

N.B. Si un produit à base solvants (acqueuse) est ajouté à une solution ne contenant que des produits à base solvants (acqueuse), la solution courante est à base solvants (acqueuse).

Si un produit à base solvants (acqueuse) est ajouté à une solution contenant des produits à base acqueuse (solvants) ou des produits dont les bases sont différentes, la solution est à bases différentes (mélange de deux bases).

Les valeurs des critères extra-techniques sans question sont les suivantes :

	min	moy	max
prix/m2	20	50	
temps de séchage	0.5	8	
nombre de produits	1	2	6
composition		mono	plus
promotion	oui	non	
base		mb	bd
facilité d'application		oui	non

avec mono, signifiant mono-composant,
plus, signifiant plusieurs composants,
mb, signifiant même base,
bd, signifiant base différentes.

La valeur moyenne du critère "promotion" est "non" car il y a très peu de produits en promotion, tandis que celle du critère "facilité d'apllication" est "oui" puisque la plupart des produits sont aisés à appliquer.

La pondération du critère "prix/m2" par rapport au critère "nombre de produits" donne :

$$\frac{2 - 1}{50 - 20} = \frac{1}{30}$$

Pour les critères dont les valeurs ne sont pas numériques, il est nécessaire de leur donner des valeurs symboliques afin de pouvoir calculer leur coefficient de pondération. Ces valeurs pourraient être par exemple les valeurs correspondantes au critère pris en référence, puisque l'objectif de la pondération est de ramener tous les critères à l'unité du critère pris comme référence. Dans ce cas, le coefficient de pondération de ces critères serait également égal à un.

Bien que cette méthode de calcul est celle que nous avons adoptée, il en existe probablement d'autres, plus sophistiquées et plus performantes.

G. L'EVALUATION D'UNE SOLUTION CANDIDATE

Reprenons la forme générale de la fonction d'évaluation dans sa version définitive. Deux cas sont à distinguer : l'évaluation porte sur une solution partielle ou terminale.

1) La valeur de la fonction d'évaluation pour une solution partielle est donnée par

$$\sum_{i=1}^{n} (vex_i + vim_i) * p_i * c_i$$

avec i, l'indice d'un critère extra-technique sans question
vex_i, la valeur explicite pour le critère i

$vim_i = \max_k \min$ produits de classe k Val_{ij}

k, l'indice de la classe k associée à un critère technique qu'il reste à satisfaire
j, l'indice d'un traitement
Val_{ij}, la valeur du critère extra-technique i pour le traitement j
p_i et c_i, respectivement les coefficients de pondération et d'importance au critère i.

2) La valeur de la fonction d'évaluation pour une solution terminale est donnée par

$$\sum_{i=1}^{n} vex_i * p_i * c_i$$

avec i, l'indice d'un critère extra-technique sans question
vex_i, la valeur explicite pour le critère i
p_i et c_i, respectivement les coefficients de pondération et d'importance

Une solution terminale ne nécessite plus aucun traitement supplémentaire. La valeur de la fonction d'évaluation pour une telle solution ne comprend donc pas de valeur implicite.

Une remarque est à faire en ce qui concerne le calcul de la valeur vex_i. En effet, tout système de peinture doit contenir au minimum 3

couches afin d'assurer une protection suffisante au support. De sorte que si une solution terminale ne comprend qu'un seul produit, il est nécessaire d'appliquer 3 couches de ce produit. Le prix au mètre carré doit dès lors être multiplié par 3 et le temps de séchage maximum doit être multiplié par 2, auquel le temps de séchage minimum (pour la dernière couche) est additionné. De même, si la solution terminale ne comprend que 2 produits (un pour la couche de fond et un pour la couche de finition), 2 couches de fond sont nécessaires. Le prix au mètre carré et le temps de séchage maximum du produit servant de couche de fond doivent être multipliés par 2.

Soit S = (T), une solution terminale contenant un seul produit T. Pour cette solution,

$$vex_{prix/m2} = 3 * pr_T$$

avec pr, le prix de T par mètre carré

et $$vex_{temps\ de\ séchage} = (2 * TSmax_T) + TSmin_T$$

avec $TSmax_T$, le temps de séchage maximum de T

$TSmin_T$, le temps de séchage minimum de T

Soit S = (T1,T2), une solution terminale contenant deux produits T1 et T2. T1 est la couche de fond et T2 est la couche de finition. Pour cette solution,

$$vex_{prix/m2} = (2 * pr_{T1}) + pr_{T2}$$

avec pr_{T1} et pr_{T2}, les prix au mètre carré de T1 et T2

et $$vex_{temps\ de\ séchage} = (2 * TSmax_{T1}) + TSmin_{T2}$$

avec $TSmax_{T1}$, le temps de séchage maximum de T1

$TSmin_{T2}$, le temps de séchage minimum de T2.

IV.4.2.2.2. Prise en compte des critères extra-techniques avec question

A la fin de la recherche (situation finale), le nombre de solutions terminales peut être strictement supérieur au nombre de solutions désirées par l'utilisateur. Dans ce cas, un choix parmi ces solutions s'avère nécessaire.

Rappelons que par définition, toute solution terminale satisfait les critères techniques et les critères extra-techniques sans question. L'ensemble de ces critères n'ayant pas été suffisants pour départager les solutions terminales, une deuxième sélection doit maintenant se faire grâce aux critères extra-techniques avec question (critères de préférence : qualité, période de l'année, mode d'application, ancienneté du produit, fonctions supplémentaires, base, odeur). L'intervention de l'utilisateur est nécessaire pour faire connaître ses préférences et pour guider le système à faire un choix parmi les solutions terminales retenues. Cette intervention se matérialise par les réponses qu'il fournira à un questionnaire.

A chaque critère extra-technique avec question est associée une question à laquelle l'utilisateur répond par "oui" ou par"non". Il est en effet important que la réponse ne comporte pas plus de deux mots afin de ne pas demander trop d'efforts de la part de l'utilisateur.

Exemple : "L'odeur vous dérange-t-elle ?"
"Désirez-vous utiliser un pistolet ?"

Le questionnaire portant sur les critères extra-techniques avec question doit être mené judicieusement. En effet, il est important

- de poser le moins possible de questions afin de ne pas ennuyer l'utilisateur. Seules les questions utiles c'est-à-dire celles qui permettent de revaloriser ou dévaloriser certaines solutions, sont posées.

- d'arriver au plus vite au nombre de solutions désirées. Les questions les plus discriminantes, c'est-à-dire celles qui permettent d'amener la plus grande distinction entre les solutions sont posées en premier lieu.

Face à la réponse de l'utilisateur relative à une question, deux situations sont observables : pour toute solution terminale

- cette solution est favorable à la réponse et son évaluation n'est pas modifiée

- cette solution est défavorable à la réponse et son évaluation est augmentée de I * P afin de la déprécier,

avec I, le coefficient d'importance du critère associé à la question,

P, le coefficient de pondération du critère associé à la question.

Exemple : si à la question "L'odeur vous dérange-t-elle ?",

l'utilisateur répond "oui", les solutions comprenant un (des) produit(s) odorant(s) voi(en)t leur évaluation augmenter de Ii * Pi avec i, l'indice du critère extra-technique (avec question) "odeur". L'évaluation des solutions non odorantes reste inchangée.

Afin de déterminer quel est le questionnaire à adresser à l'utilisateur, nous allons procéder en trois temps. Tout d'abord, nous donnons quelques définitions et des exemples nous permettant de voir quels sont les aspects à considérer dans l'élaboration du questionnaire. Ensuite, nous proposons un algorithme qui détermine quelles sont les questions à poser en tenant compte des aspects vus précédemment, qui enregistre les réponses et qui, en fonction de celles-ci, choisit parmi les solutions terminales les solutions préférables pour l'utilisateur. Enfin, connaissant les questions qu'il faut soumettre à l'utilisateur, nous étudions comment il est préférable de les formuler.

A. DEFINITIONS ET EXEMPLES

Les questions discriminantes sont les questions qui portent sur les critères extra-techniques avec question qui permettent de faire une distinction entre les solutions, c'est-à-dire sur les critères dont les valeurs dans chaque solution ne sont pas identiques.

Exemple.

Soit S1, S2, S3, S4, quatre solutions terminales,
1, 2, 3, 4, quatre critères extra-techniques avec question,
A, B, C, les valeurs que peuvent prendre ces critères.

	S1	S2	S3	S4
1	A	B	A	A
2	C	C	C	C
3	B	B	A	C
4	A	A	A	A

Les questions se rapportant aux critères 1 et 3 sont discriminantes et les questions se rapportant aux critères 2 et 4 ne le sont pas, elle ne permettront pas de distinguer les solutions terminales.

Partons d'un exemple pour illustrer les aspects à considérer afin de mener à bien le questionnaire.

Soit quatre solutions terminales S1, S2, S3, S4 et S5 dégagées par la recherche heuristique, avec les évaluations suivantes :

S1 : 9 S2 : 15 S3 : 16 S4 : 17 S5 : 22.

Soit X = 2, le nombre de solutions désirées par l'utilisateur.

Le but du questionnnaire et donc le problème qui nous concerne est de savoir si l'ordre établi selon l'"évaluation" des solutions, ne doit pas être modifié à cause des critères extra-techniques avec question. Pour ce faire, le premier problème est de connaître quelles sont les solutions dont l'ordre est susceptible d'être modifié en fonction des préférences de l'utilisateur. Le deuxième problème est de savoir quel est le nouvel ordre de ces solutions et comment il peut être obtenu. A l'issue de ces deux points, nous pourrons déterminer quelles sont les meilleures solutions.

1) Le premier problème est de calculer s'il existe des solutions dont le numéro d'ordre ne peut pas être modifié, quelles que soient les préférences de l'utilisateur. Dans notre exemple, l'écart entre 9 et 15 étant assez grand, il faut s'assurer que même si tous les critères extra-techniques avec question de la première solution sont défavorables par rapport aux préférences de l'utilisateur, cette solution est quand même la meilleure.

C'est le cas si $9 + \sum_{i} (Ii * Pi) <= 15$

avec i, l'indice d'un critère extra-technique avec question à considérer (utile).

Supposons que $\sum_{i} (Ii * Pi) = 5$.

Dès lors, la première solution ne peut être remise en cause et elle est désormais considérée comme étant la meilleure.

Il reste à trouver quelle est l'autre solution convenant le mieux au problème de l'utilisateur.

De façon générale, l'ensemble des solutions peut donc être décomposé en 2 sous-ensembles complémentaires : les solutions en compétition et les solutions hors compétition.

Les <u>solutions hors compétition</u> sont les solutions dont l'ordre de préférence ne peut être remis en cause à l'issue des réponses aux questions posées à l'utilisateur. Elles sont soit définitivement bonnes, soit définitivement mauvaises.

Les autres solutions sont les <u>solutions en compétition</u>.

Dans notre exemple, les solutions en compétition sont S2, S3, S4 et leur ordre peut être remis en cause puisque la différence entre chacune de leur "évaluation" est strictement inférieure à 5 (c'est-à-dire $\sum_i Ii * Pi$)

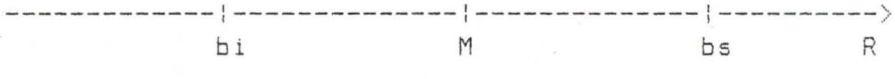
Nous pouvons déduire que de manière générale, l'ensemble des solutions en compétition pour lesquelles des questions sont nécessaires, est l'ensemble des solutions dont l'"évaluation" est comprise dans l'intervalle

$$\left[M - \sum_i (Ii * Pi) \quad , \quad M + \sum_i (Ii * Pi) \right]$$

avec i, l'indice d'un critère extra-technique avec question à considérer ,
M, l'évaluation de la Xième solution terminale,
X, le nombre de solutions désirées par l'utilisateur
et où les solutions terminales sont triées par ordre croissant sur leur "évaluation".

Les solutions dont l'"évaluation" se situe en dehors de cet intervalle sont hors compétition.

Pour faciliter la compréhension de cette définition, représentons M, la borne inférieure bi et la borne supérieure bs de l'intervalle sur l'axe des réels.



Une solution I dont l'"évaluation" est strictement inférieure à bi est définitivement bonne car même si toutes les réponses de l'utilisateur sont défavorables à cette solution, la valeur de la fonction d'évaluation pour I ne peut être augmentée de plus $\sum_i Ii * Pi$.

Son "évaluation" ne pourra pas être supérieure à M et de ce fait, I fait partie des X meilleures solutions.

De la même manière, une solution S dont l'"évaluation" est strictement supérieure à bs est définitivement mauvaise car même si toutes les réponses de l'utilisateur sont favorables pour cette solution (dans ce cas, la valeur de la fonction d'évaluation est inchangée), les valeurs de la fonction d'évaluation pour les solutions de l'intervalle [bi, M] ne peuvent être augmentées de plus de $\sum_i (Ii * Pi)$ et ne pourront donc jamais être supérieures à bs. La solution S ne pourra donc jamais être meilleure qu'une solution dont l'"évaluation" est inférieure ou égale à M. Elle ne pourra donc jamais être parmi les X premières solutions.

Les solutions dont l'"évaluation" se trouve dans l'intervalle peuvent voir leur "évaluation" varier de $\sum_i (Ii * Pi)$. Leur ordre peut donc se trouver modifié et de ce fait, les X premières solutions peuvent varier.

Dans notre exemple :

$\sum_i (Ii * Pi) = 5$
X = 2
M = l'évaluation de la 2ième solution terminale
= 15.

L'intervalle à considérer est donc [10, 20] et les solutions en compétition sont S2, S3, S4.

Les solutions situées en dehors de cet intervalle ne peuvent être remises en question : elles sont définitivement bonnes ou définitivement mauvaises.

Cet intervalle est à considérer si et seulement si :

$$\sum_i (Ii * Pi) >= N - M$$

avec N, la valeur de la fonction d'évaluation pour la (X + 1)ième solution.

Exemple.

Soient 5 solutions terminales et leur "évaluation" respective,

S1 : 9 S2 : 12 S3 : 15 S4 : 21 S5 : 22,
X = 3,
$\sum_i (Ii * Pi) = 5$,
M = 15,
N = 21 et l'intervalle est [10, 20].

N - M = 6 > 5, de sorte que toutes les solution situées avant S3 ne pourront jamais être moins bonnes que les solutions situées après S4. En effet, leur "évaluation" peut au plus être augmentée de 5, ce qui n'est pas suffisant pour être supérieur à l'"évaluation" de S4. Rappelons qu'une solution favorable aux réponses de l'utilisateur ne voit pas son "évaluation" diminuer, elle reste constante. Donc au mieux, l'"évaluation" de S4 reste égale à 21.

2) Le deuxième problème est de savoir quelle question poser en premier lieu afin de découvrir le plus vite possible les meilleures solutions.

La première question à poser est bien sûr la question la plus discriminante, c'est-à-dire celle amenant le plus de distinction entre les solutions et donc celle qui a la valeur I * P la plus grande, avec I et P les coefficients d'importance et de pondération du critère qui est associé à cette question.

Les solutions défavorables à la réponse de cette question voient leur évaluation augmenter de I * P. Cela a pour effet de rendre un peu meilleures certaines solutions, et un peu moins bonnes les autres.

B. ALGORITHME DE RECHERCHE

- Situation initiale

* Nous disposons de S solutions terminales, parmi lesquelles X sont désirées par l'utilisateur (avec X < S), ces S solutions sont triées par ordre croissant sur leur évaluation

* Les questions discriminantes sont recherchées.

* L'ensemble des solutions en compétition est calculé grâce à l'intervalle $[\ M - \sum_i (Ii * Pi)\ ,\ M + \sum_i (Ii * Pi)\ ]$

avec i, indice d'un critère correspondant à une question discriminante,
Ii et Pi, les coefficients d'importance et de pondération du critère i,
M, l'évaluation de la Xième solution.

Les solutions dont l'"évaluation" est inférieure à la borne inférieure de cet intervalle sont considérées comme étant définitivement bonnes. Les solutions dont l'"évaluation" est supérieure à la borne supérieure de cet intervalle sont considérées comme définitivement mauvaises.

- Situation générale

* Nous disposons d'un ensemble de solutions et d'un ensemble de questions à poser.

* La question dont le produit des coefficients d'importance et de pondération est le plus élevé, soit la question portant sur le critère i, est posée et la réponse de l'utilisateur est

enregistrée; cette question est supprimée de l'ensemble des questions à poser.

* L'"évaluation" des solutions défavorables à la réponse de l'utilisateur est augmentée de Ii * Pi

* M est réajusté.

- Situation finale

Plusieurs cas peuvent provoquer l'arrêt de la recherche.

* Il n'existe plus de questions à poser. L'utilisateur a fait connaître ses préférences pour tous les critères qui peuvent amener une distinction entre les solutions en compétition.

* Des questions supplémentaires ne sont pas nécessaires car

$$N - M > \sum_{j} (Ij * Pj)$$

N étant la valeur de la fonction d'évaluation pour la (X + 1)ième solution

et j représentant l'indice d'un critère qu'il reste à considérer (ou d'une question discriminante qu'il reste à poser).

A ce moment, les moins bonnes solutions ne pourront plus devenir meilleures que les bonnes solutions.

Les X premières solutions dans l'ordre de leur "évaluation" sont les solutions les mieux adaptées au problème de l'utilisateur en accord avec les critères techniques et les critères extra-techniques (avec et sans question). Ces systèmes-solution sont visualisés à l'écran pour en faire part à l'utilisateur.

C. FORMULATION DES QUESTIONS

Deux aspects sont à considérer en ce qui concerne la formulation des questions.

- Les questions doivent être formulées de manière à ce que l'utilisateur comprenne rapidement ce dont il s'agit et qu'il n'éprouve aucune difficulté à y apporter sa réponse. La terminologie doit donc être adaptée au type d'utilisateur. Cependant, il s'agit d'un vocabulaire assez simple puisque les critères extra-techniques avec question ne sont pas des critères à termes techniques. Ils se réfèrent à des spécificités assez courantes de la peinture.

- D'un point de vue programmation, il est nécessaire de procéder toujours de la même manière pour une réponse, quelle que soit la question posée.

  La question Q doit être formulée de façon à ce que, étant donné une valeur de V :

  * si le système pose la question Q à l'utilisateur et si celui-ci répond par l'affirmative, les solutions dont la valeur du critère considéré est V, voient leur "évaluation" augmenter du produit des coefficients d'importance et de pondération de ce critère.

  * si l'utilisateur répond par la négative, les "évaluations" de toutes les solutions restent inchangées.

  La relations entre la question Q et la valeur V peut être formalisée de la manière suivante :

  ((question, Q), (valeur, V)).

Exemple.

  Soit la question
  ((question, "L'odeur vous dérange-t-elle ?"), (valeur, oui)).

  Si l'utilisateur répond par l'affirmative, les solutions qui comportent des produits odorants doivent être pénalisées par une augmentation de leur "évaluation" respective. De telles solutions sont celles dont la valeur du critère "odeur" est "oui".

  Si l'utilisateur répond par la négative, cela signifie qu'il est indifférent au fait que le système de peinture est odorant ou non. Dans ce cas, aucune solution n'est à pénaliser.

IV.4.3. Amélioration de la recherche

Un être intelligent chargé de résoudre un problème peut améliorer son efficacité de différentes manières [HAY84] :

- il possède les connaissances qu'il utilise souvent, évite les erreurs, et fait des distinctions selon le type d'application.

- il élimine rapidement les chemins de recherche qui s'avèrent inutiles. Il se taille des chemins en supprimant les classes infructueuses. Le coût de la résolution d'un problème doit satisfaire certaines limites. Habituellement, ces limites réduisent le temps et l'espace disponibles à la résolution. Ce qui conduit à la limitation du nombre de solutions partielles à considérer.

- il élimine toute redondance en calculant une seule fois ce dont il a besoin à plusieurs reprises.

C'est ce que nous avons tenté de réaliser.

En ce qui concerne plus particulièrement le dernier point, il nous a semblé intéressant de séparer la recherche des préparations de la recherche des traitements. En effet, lorsque l'utilisateur désire plusieurs solutions, le système expert étudie les différentes suites de traitements et de préparations permettant de faire la transition entre l'état initial et l'état final du support. Or, contrairement aux traitements, la suite des préparations pour un même problème est unique. Cela se comprend quand on sait que d'une part, à chaque dégradation correspond une seule préparation qui peut y remédier et que d'autre part, les préparations doivent se succéder selon un ordre bien précis. De sorte qu'il est inutile de faire la recherche des préparations pour chaque solution demandée par l'utilisateur. La séparation des deux recherches peut se faire en décomposant le problème en deux parties :

- la recherche d'une solution permettant de passer de l'état final du support à l'état initial dont l'état de surface est "recouvrable" et donc qui ne nécessite pas de préparation
- la recherche d'une solution permettant de passer de l'état initial recouvrable à l'état initial du support.

Seule la première partie peut conduire à plusieurs solutions, la deuxième étant composée de préparations uniquement.

Cette démarche est d'autant plus intéressante que la recherche d'une suite de préparations est très longue car pour chacune d'elles, il existe beaucoup de règles à exécuter pour vérifier leurs conditions d'applicabilité.

D'un autre côté, ayant choisi de conserver la démarche utilisée dans [DEC86], nous n'avons pas éliminé la redondance suivante : lorsque l'utilisateur a donné la valeur de l'attribut "nature" du support, le programme de recherche pourrait ne considérer que les produits de cette "nature" au lieu de vérifier à chaque sélection d'un produit, que sa "nature" est compatible avec celle du support. Afin de ne travailler que sur une partie des produits, un marquage est utile. Un tel marquage s'est d'ailleurs révélé très gourmant en place mémoire qui pour l'instant fait défaut.

IV.5. AVANTAGES DE L'IMPLEMENTATION SUR ORDINATEUR DU CHOIX D'UN SYSTEME DE PEINTURE COMPRENANT LES CRITERES COMMERCIAUX

L'avantage de l'implémentation sur ordinateur d'un tel système est d'uniformiser le conseil peinture. En effet, dans le conseil peinture actuel, selon la personne à qui s'adresse le client, la solution obtenue diffère, bien qu'elle soit techniquement correcte et conforme au problème posé par le client. Ceci s'explique par le fait qu'à un problème posé, peuvent correspondre plusieurs systèmes de peinture corrects et que le choix parmi ceux-ci est guidé par des critères commerciaux mais aussi par des critères qui découlent de la personnalité du spécialiste en peinture : la préférence de certains

produits selon son expérience, ... Il s'agit en fait de critères subjectifs.

L'ordinateur permet donc d'avoir une source d'informations UNIQUE et commune à toute l'entreprise, en éliminant notamment le caractère émotif de la personne qui conseille. Il est alors important que les représentants et le personnel de l'entreprise n'aient accès qu'à une ou deux solutions, lesquelles sont considérées comme étant les plus favorables aux désirs du client et de la firme.

Le second interêt de l'utilisation d'un ordinateur est que l'on est certain de prendre en considération à tout moment TOUTES les informations dont on dispose pour choisir une suite de traitements résolvant un problème de mise en peinture. L'ordinateur possède également les renseignements relatifs aux critères commerciaux que l'utilisateur a pu fournir, et durant la recherche d'une suite de traitements, il tient compte d'absolument toutes ces informations. Ce qui n'est pas toujours le cas lorsqu'il s'agit d'une personne humaine pour qui un oubli est chose courante.

V. LE PROCESSUS D'EXPLICATION DU RAISONNEMENT DU SYSTEME EXPERT

Le but de ce chapitre est de proposer une technique d'explication permettant de fournir à l'utilisateur la justification des résultats du système expert et de lui donner la possibilité de poser des questions. L'introduction présente en toute généralité en quoi consiste la production d'explications, pourquoi elle est nécessaire et les propriétés qu'elle doit satisfaire. Afin de savoir ce qu'il faut expliquer pour le système Conseil, il est nécessaire d'étudier quels sont les types d'explications à fournir en fonction du type de l'utilisateur. A partir de ces renseignements, nous pourrons déduire les questions auxquelles le système expert devra répondre et les types de réponses qu'il devra y apporter. Le dernier point abordera la stratégie à utiliser pour que le système expert soit capable de donner, pour chaque question, des réponses cohérentes et complètes.

V.1. INTRODUCTION

V.1.1. Le concept d'explication

1. Définition générale

Pour être acceptable, un système expert doit être capable de fournir des explications compréhensibles sur les actions qu'il a effectuées et les stratégies qu'il a employées et leur justification. Ces explications décrivent le plan que le programme a suivi pour atteindre une solution.

Cependant, le contenu des explications dépend de plusieurs facteurs, que nous allons tenter de découvrir par les aspects décrits ci-dessous.

2. Le besoin d'explication [BUC84]

La capacité d'expliquer le raisonnement suivi est habituellement considéré comme un élément important d'un système expert, et ceci pour plusieurs raisons :

- compréhension du raisonnement
- correction (mise au point) du système expert
- apprentissage de la méthode de résolution
- acceptation des résultats fournis par le système expert
- persuasion de l'exactitude des résultats fournis par le système expert

COMPREHENSION

Comprendre le contenu de la base de connaissances et les lignes du raisonnement suivi, est un des buts principaux de l'explication. Le concepteur du système et l'utilisateur ont besoin de comprendre les connaissances contenues dans le système, dans le but de les mettre à jour et de les utiliser efficacement.

Le système peut parfois prendre l'initiative d'informer les utilisateurs sur les étapes du raisonnement qu'il effectue (en donnant notamment des conclusions intermédiaires lui permettant d'atteindre la conclusion finale). Habituellement, le système fournit les explications correspondant aux requêtes spécifiques formulées par l'utilisateur.

CORRECTION

Cet aspect est important, d'autant plus que les bases de connaissances sont construites progressivement, incrémentalement ce qui risque de mettre en cause les caractères de cohérence et de complétude. En étant capable de donner à l'utilisateur le raisonnement effectué pour atteindre une solution, le système expert permet au programmeur et à l'expert de vérifier si tous les aspects du problème et toutes les règles à appliquer ont été pris en considération. En cas d'erreur ou d'incomplétude, ils peuvent, grâce à l'explication, localiser l'endroit où une modification s'avère utile.

APPRENTISSAGE

L'apprentissage est une autre raison importante pour fournir un aperçu de la base de connaissances. Afin que l'utilisateur puisse apprendre quelque chose du système expert, il est nécessaire de rendre compréhensibles la base de connaissances et les lignes du raisonnement. L'utilisateur naïf peut alors enrichir ses connaissances en ce qui concerne le genre de problème traité par le système expert.

ACCEPTATION

L'acceptation et la persuasion sont étroitement liées. Rendre un système expert acceptable est en partie convaincre des utilisateurs que ses conclusions sont rationnelles. S'ils comprennent comment le système atteint des résultats pour plusieurs problèmes, c'est-à-dire s'ils connaissent la (les) stratégie(s) sur lesquelles le système s'est basé, et s'ils pensent que le processus suivi est raisonnable, ils feront probablement confiance aux conclusions trouvées pour d'autres problèmes. Pour la même raison, il est important de montrer que le système tient compte des données particulières d'un problème et donc de distinctions spécifiques à différents cas.

PERSUASION

Persuader les utilisateurs que les conclusions du système sont correctes, nécessite également une certaine vue de la base de connaissances et des lignes du raisonnement. Quand une personne consulte un expert conseil, elle s'attend à comprendre suffisamment ses conclusions (et les bases de celles-ci) que pour accepter la responsablilité d'agir en fonction de ces conclusions. Dans le domaine médical, par exemple, les médecins ont une responsabilité morale et légale des conséquences de leurs actions. Dès lors, ils doivent comprendre pourquoi - et parfois en être persuadés que - les conseils d'un expert sont appropriés.

Nous pouvons constater dés à présent que le concept d'explication se définit de manière différente selon le type d'utilisateur et l'objectif recherché (ces deux éléments étant fortement liés).

V.1.2. Les trois éléments structurant un système d'explication [WAR83]

La production d'explications peut être considérée comme structurée à partir de trois éléments principaux : l'aspect épistémologique, l'aspect utilisateur et l'aspect rhétorique.

1. Aspect épistémologique.

Le fondement de toute explication est de décrire un modèle des connaissances et du processus de raisonnement. Le travail d'explication que nous caractérisons d'épistémologique, concerne la connaissance requise pour résoudre un problème (pouvant être soumis au système expert) et les aspects du comportement du résolveur de problème qui doivent être expliqués. Dans l'essai d'imiter l'être humain pour la résolution de problèmes, les chercheurs pensent que les modèles existants du raisonnement humain sont trop limités pour permettre une résolution et une explication robuste. Un aspect clé de la recherche dans ce domaine est donc l'étude et la formalisation du processus de raisonnement, et la manière dont la structure de connaissances est manipulée.

Se pose donc le problème de choix de modélisation et de formalisation du processus de raisonnement dans un ordinateur. Ce problème fut étudié par plusieurs chercheurs qui optèrent vers les solutions suivantes :

- Shortliffe et Davis utilisent une structure de buts et de règles d'inférence pour conduire une consultation médicale. La traduction de ces règles constitue l'explication du processus d'inférence.

- Clancey a analysé le problème de la représentation de chaque type

de connaissance, séparément et de manière explicite dans le but de les communiquer clairement à l'utilisateur naïf.

- Swartout utilise des principes et des contraintes propres au domaine concerné pour former le processus de raisonnement.

Dans chaque cas, la représentation des connaissances et du raisonnement doit se faire selon un formalisme qui peut être utilisé pour résoudre des problèmes, et ensuite pour justifier les actions du programme.

Ces problèmes seront traités au point V.5.

2. Aspect utilisateur

Etant donné une idée des connaissances nécessaires pour résoudre un problème et une structure de représentation, il faut encore déterminer ce qui doit être expliqué à une personne particulière. Le principe de base est de générer des explications qui tiennent compte des connaissances de l'utilisateur et de ses préférences. Selon que l'utilisateur est la personne chargée de corriger le programme ou une personne naïve ne connaissant que quelques notions du domaine traité, le contenu et le niveau de détail de l'explication seront totalement différents.

Cet aspect sera traité au point V.2.

3. Aspect rhétorique

Dès que le contenu d'une explication a été déterminé, il reste le problème de savoir comment présenter l'explication à l'utilisateur. L'aspect rhétorique se préoccupe de fournir une explication qui soit compréhensible. Il est donc important de savoir quelles sont les limites de l'utilisateur, sa capacité d'assimiler de nouvelles informations et son niveau de connaissance. Ces éléments déterminent le vocabulaire à utiliser et la forme des explications (graphique, texte, plan, ...). Le système doit donc être capable de décrire la même information de différentes manières selon la personne à qui les explications sont destinées.

Tout processus d'explication implique également un certain degré d'interactivité avec l'utilisateur. Ce degré dépend du domaine d'application et de l'étendue des explications fournies.

Dans le cas où la requête d'explication se fait sous forme de questions, il est bien sûr vital de comprendre ces questions, de savoir ce que désire l'utilisateur et de fournir une réponse correcte.

Ces problèmes de langage naturel et de forme d'explication seront repris dans le point V.2. (terminologie).

V.1.3. Propriétés des explications à fournir [WAR83]

De bonnes explications doivent respecter dans la mesure du possible, les critères développés ci-dessous.

- Il est important que la capacité d'explication du système puisse traiter des questions au sujet de tous les aspects pertinents de la connaissance et des actions du système. Il doit donc être capable d'éclairer l'utilisateur sur les points suivants :

    * comment une décision a été prise
    * comment une information a été utilisée
    * pourquoi une information n'a pas été utilisée
    * pourquoi une telle décision n'a pas été prise
    * comment une certaine information a été découverte.

- Les explications ne doivent présupposer aucune population particulière d'utilisateurs. En effet, les utilisateurs étant de type divers, les techniques qui génèrent les explications doivent être suffisamment flexibles pour adapter le modèle à l'utilisateur. Le vocabulaire habituel de ces types de personnes étant différent, le système doit pouvoir s'accommoder au choix de la terminologie afin d'être compréhensible.

- Les explications doivent être éloquentes et complètes : le numéro ou le nom de la règle appliquée ne suffit pas. Les explications générées doivent fournir une image assez précise des étapes qui ont été réalisées.

- Les explications peuvent être concrètes ou abstraites. Des explications concrètes font référence à des aspects propres au problème traité.
  Par contre, des explications abstraites s'articulent autour de principes généraux qui peuvent être appliqués dans différentes situations. De telles explications sont utiles lors d'apprentissages et d'explications par analogie.

  Exemple dans le domaine de la peinture.

    - Explication concrète : "la peinture Aubexol assure une résistance contre l'abrasion forte. Cette peinture résiste donc également contre l'abrasion légère et dès lors convient à votre problème".

    - Explication abstraite : "une peinture assurant une certaine résistance, assure également toute résistance moins exigeante".

  Les explications doivent être concrètes ou abstraites selon la situation. Il doit donc être possible de fournir les deux formes, ceci afin de faciliter la compréhension, et de la stratégie, et de la façon dont elle est appliquée à un problème particulier.

- Les explications peuvent être enrichies sur demande mais elles sont

développées tout au plus jusqu'au niveau de détail correspondant à l'intérêt et à l'aptitude maximum de l'utilisateur.

Le système d'explication doit comprendre plusieurs niveaux de détail. Le premier niveau est caractérisé par des explications très générales, le deuxième niveau par des explications un peu moins générales, et ainsi de suite jusqu'au dernier niveau qui correspond aux détails d'implémentation. Il s'agit du niveau de détail le plus élevé.

Toute explication fournie à l'utilisateur commence par le premier niveau. Si ces explications ne sont pas assez explicites, il peut demander de les préciser en vue d'obtenir des détails supplémentaires. Il s'agit alors d'explications du niveau supérieur. Ce processus d'enrichissement d'explication peut se répéter autant de fois qu'il existe de niveaux de détail pour chaque type d'utilisateur.

* Le niveau doit être suffisamment élevé pour que les opérations citées soient compréhensibles. Des phrases trop générales risquent de ne pas éclairer du tout l'utilisateur. Des détails supplémentaires sont dans ce cas nécessaires.

  Exemple : "cette décision résulte du meilleur choix."
  Cette justification explique très peu de choses voire rien, puisqu'elle ne révèle pas les critères de choix pris en considération.

* Le niveau de détail doit également être suffisamment bas pour que les opérations citées soient significatives et en même temps débarassées de détails superflus.

Le passage d'un niveau de détail déterminé vers le niveau de détail souhaité, peut se faire grâce à un processus dynamique descendant/ascendant, c'est-à-dire grâce à un processus de ZOOMING. En effet, l'utilisateur peut demander de développer une explication par un système de "pourquoi successifs", chaque "pourquoi" lui permettant d'accéder au niveau de détail suivant. De même, il peut accéder au niveau inférieur (qui comprend moins de détails) grâce à un système de "comment successifs".

Exemple.

Soit S.E., le système expert,
U., l'utilisateur.

Les dialogues qu'ils échangent pourraient être les suivants :
"U. : Pourquoi la peinture PERMALINE a-t-elle été choisie ?
S.E. : PERMALINE permet de se rapprocher de la solution et elle convient à votre problème.
(niveau 1)
U. : Pourquoi ?

S.E. : PERMALINE donne l'aspect brillant que vous avez demandé et cette peinture respecte l'état du support : elle est adaptée pour le bois et pour l'environnement urbain. Elle est également compatible avec la couche inférieure qui est MULTIPRIMER.
(niveau 2)
U. : Pourquoi est-elle compatible avec MULTIPRIMER ?
S.E. : PERMALINE et MULTIPRIMER sont compatibles car
- leurs liants sont compatibles
- leurs solvants sont compatibles
- leurs pigments sont compatibles
(niveau 3).

On comprend aisément que l'utilisateur naïf n'a pas accès aux niveaux supérieurs à 2, ceux-ci comportent en effet trop de détails techniques qui sont hors de portée de sa connaissance et de son intérêt.

Pour permettre la correction du programme, le système doit se placer au niveau le plus détaillé, c'est-à-dire au niveau de la règle (dernier niveau). Des explications plus générales peuvent également être générées en omettant les détails qui concernent les niveaux plus détaillés.

- Le processus d'explication doit être aisé à utiliser. Une personne novice doit être capable de demander des explications sans pour autant passer de longs moments à apprendre comment former des requêtes.

V.2. TYPES D'EXPLICATIONS SELON LE TYPE D'UTILISATEUR

Dans le cadre du système Conseil développé pour la firme Trimetal, seules des consultations non-interactives ont lieu. En effet, l'utilisateur n'a la possibilité d'interroger l'ordinateur qu'APRES le moment où celui-ci communique le(s) système(s) de peinture correspondant le mieux au problème posé. La capacité d'explication pour un tel système est donc limitée à des questions qui portent uniquement sur l'état final des connaissances, sur la solution proposée et sur la manière dont cette solution a été obtenue.

Dans cette application, l'utilisateur n'intervient au cours de la recherche que lorsque le système a besoin de son avis afin de mieux spécifier et particulariser le problème à résoudre (cfr IV.4.2.2.2). Outre cette éventuelle intervention, l'utilisateur ne peut guider la recherche puisque les principes à appliquer dans le choix d'une peinture sont fixes. Lorsque le processus de recherche est terminé, l'interactivité devient complète car l'utilisateur peut interroger le système sur la justification du choix ou du rejet d'un quelconque système de peinture.

La première étape est de savoir quelles sont les questions que l'utilisateur peut poser et donc quels types d'explication le système

doit être capable de fournir.

Dans l'entreprise Trimetal, plusieurs catégories d'utilisateurs seront amenées à se servir du système Conseil dans le cadre de différentes applications. Pour chacune de celles-ci, nous allons déterminer le type d'explication à fournir c'est à dire son contenu, son niveau de détail et sa terminologie.

V.2.1. Le programmeur, le chimiste et le commercial

Après une première ébauche du système, le programmeur consacre habituellement énormément de temps (généralement plus de 50 % du temps de développement du système) à tester le système, à vérifier si les solutions proposées par l'ordinateur sont raisonnables. C'est la phase de mise au point, de correction.

Pour parvenir au mieux à cette fin, une collaboration entre programmeurs et experts (ces derniers étant, en l'occurrence, les chimistes et les commerciaux) s'avère utile.

CONTENU

Ces personnes demandent des explications au sujet du système solution produit par l'ordinateur, elles veulent savoir comment cette solution a été atteinte, quelles sont les règles et stratégies sous-jacentes. En cas de discordance entre la solution choisie et leur attente, les experts peuvent en outre proposer un système de peinture différent de celui choisi par l'ordinateur et demander pour quelles raisons il n'a pas été sélectionné.

Le processus d'explication permet donc de vérifier si leurs propres connaissances ont été interprétées dans le sens opportun et si l'ensemble des règles appliquées est complet et cohérent. En divulguant les règles qui se sont révélées pertinentes pour un problème particulier, il rend possible la découverte de celles dont le contenu est erroné ou dont les conditions d'application sont incorrectes et celles qui, à tort, sont absentes.

L'interactivité entre le système et l'utilisateur doit être suffisante pour rendre possible la découverte de telles erreurs.

Les experts ayant soumis leur opinion quant au caractère correct du système, le programmeur peut dès lors procéder aux modifications souhaitées. Eventuellement, un processus d'acquisition de connaissances pourrait être mis à la disposition des experts afin qu'ils puissent effectuer eux-mêmes les changements qu'ils jugent nécessaires. Un tel processus s'avère également utile lors de la maintenance du programme puisqu'au cours du temps, des mises-à-jour concernant les principes mêmes de la peinture peuvent se produire.

Les chimistes et le programmeur peuvent également demander au système expert de ne pas tenir compte des critères économiques et

de leur fournir uniquement les solutions techniquement correctes. Les chimistes peuvent de cette manière vérifier si les règles techniques ont bien été appliquées indépendamment des aspects commerciaux qui ne sont pas de leur ressort. Le programmeur quant à lui, peut vérifier si les principes techniques ont été programmés correctement.

TERMINOLOGIE et NIVEAU DE DETAIL

Pour la phase de mise au point, les règles ne peuvent être soumises aux experts sous la forme qu'elles revêtent au sein du programme. Les explications doivent être compréhensibles et donc traduites en un langage connu de ces utilisateurs.

Le niveau de détail doit rester suffisamment bas afin de ne pas tomber dans les astuces de programmation dont les experts n'ont que faire. Cependant, tout en restant très peu élevé, le niveau de détail peut être choisi par un système de "pourquoi successifs" qui détaillent de plus en plus les réponses fournies par l'ordinateur.

Par contre, à la première phase de développement du système expert, qui n'implique que le programmeur, doivent correspondre des explications du niveau le plus élevé possible et pouvant rester sous la forme du langage de programmation. Il pourra dès lors repérer facilement les endroits où les stratégies qu'il a communiquées ne sont pas correctement implémentées.

Le système du "pourquoi et comment successifs" lui permet de voyager allègrement dans tous les recoins du programme.

V.2.2. Le "do it yourself" ou le particulier

Ce type d'utilisateur désigne le client novice qui souhaite acquérir du matériel de peinture dans un centre de bricolage (grande surface vendant du matériel de bricolage : peinture, menuiserie, plomberie, ...).

Une distinction importante est à faire à ce niveau :

- le client se trouve sur le lieu de vente et il a accès à un ordinateur commun à toute la clientèle de cette grande surface. Dès lors, l'explication du choix du système de peinture proposé entrainerait un temps d'attente intolérable pour les autres personnes qui désirent un conseil peinture. Un processus d'explication est donc peu envisageable dans cette situation (à moins qu'il n'y ait plusieurs ordinateurs à la disposition des clients). Cela ne constitue pas un handicap majeur car le client, étant novice dans ce domaine, fait totallement confiance au système Conseil et, satisfait d'avoir une réponse à son problème, il est prêt à appliquer la solution proposée.

- le client ne se trouve pas sur le lieu de vente mais peut avoir accès par un réseau analogue au Minitel ou par télex, à un bureau central de conseil en peinture. Dans ce cas, l'accès individuel rend possible des explications complémentaires au système de peinture apparaissant à l'écran.

CONTENU

Le client a la possibilité de demander quels sont les grands principes qui ont été appliqués pour résoudre le problème qu'il a soumis. Le processus d'explication reprend alors les exigences du particulier et expose la manière dont elles ont été satisfaites (explications concrètes).

La réponse du système expert face à un problème déterminé comporte un cas particulier : "Il n'existe pas de solution à votre problème.". Le processus d'explication tentera de découvrir la justification appropriée : plusieurs exigences sont incompatibles entre elles (exemple : extérieur et mat), une ou plusieurs exigences ne peuvent être satisfaites par un produit de la gamme Trimetal, ...

L'utilisateur peut en outre demander la ou les raisons pour lesquelles un certain produit ne convient pas pour résoudre son problème. L'ordinateur lui expliquera alors ce qui a provoqué son rejet.

Ce type d'utilisateur, étant généralement novice dans le domaine de la peinture, n'éprouvera pas le besoin de connaître la justification du rejet d'un système complet, puisqu'il est incapable de proposer un tel système.

Un point sur lequel le client désire souvent des informations complémentaires, est le mode d'emploi et les conditions d'application de la peinture qu'il doit utiliser. Ces informations reprennent la notion de prix, de temps de séchage, de durabilité, du matériel à employer, ..., c'est-à-dire tous les critères extra-techniques. Toutefois, ce genre d'explication n'a aucune raison d'être lorsqu'on se trouve sur le lieu de vente puisqu'elles sont disponibles sur les boîtes de peinture correspondantes.

Des explications supplémentaires et indépendantes du problème soumis par le client, pourraient être fournies : qu'est-ce qu'une peinture, quels sont les grands principes à respecter dans le domaine de la peinture, pourquoi deux éléments déterminés sont-ils incompatibles entre eux, ... Il s'agit d'explications abstraites.

TERMINOLOGIE

Les explications destinées aux particuliers sont exprimées en des termes très simples. Elles ne peuvent comprendre des concepts techniques qui ne feraient qu'obscurcir leur esprit. Dans le cas

éventuel où un tel système se trouverait sur le lieu de vente, les phrases devront être très simples et très courtes, d'une part pour minimiser le temps d'occupation du terminal, et d'autre part pour ne pas risquer d'ennuyer le client.

NIVEAU DE DETAIL

Les explications à fournir doivent se situer à un niveau de détail très peu élevé. Des explications trop détaillées et donc très techniques ne seraient pas comprises par la plupart des utilisateurs, de sorte qu'il est préférable de parler principalement des aspects extérieurs, visibles à l'oeil nu, tels la brillance, la dureté, ... Un seul niveau de détail est suffisant. Il faut tenir compte du fait que la peinture est une "corvée" et qu'il ne faut pas trop accabler le client.

V.2.3. Le peintre professionnel

CONTENU

Si le peintre professionnel est amené à employer un tel système, c'est dans le but d'obtenir une confirmation par rapport à la solution qu'il préconise. Une explication n'est alors requise que dans le cas où le système proposé comprend un produit nouveau que le peintre ne connaît pas encore très bien. Dans le but de faire connaître les nouveaux produits, il peut avoir accès à la fiche technique d'une peinture déterminée. Cet aspect est important dans la mesure où ce type de personne est assez réticent vis-à-vis des produits auxquels il n'est pas habitué.

Si la solution proposée par le système ne comprend que des produits traditionnels, aucune explication n'est nécessaire puisque le peintre connaît les principes à appliquer pour construire tout système de peinture. Il est en effet préférable de ne pas lui expliquer ce dont il a une parfaite maîtrise.

Outre les questions que le client particulier peut poser, le professionnel a la possibilité de proposer le système qu'il préconisait, afin de connaître la justification de son rejet.

TERMINOLOGIE

Le professionnel a une formation assez complète, ce qui signifie que les explications peuvent comprendre certains détails techniques.

NIVEAU DE DETAIL

Il est nécessaire de tenir compte du fait que ce type d'utilisateur connaît un certain nombre de choses dans le domaine de la peinture, de sorte que nous devons éviter autant que possible de lui répéter les éléments qu'il connait déjà. Pour atteindre cet objectif, le peintre choisit lui-même le niveau de détail qu'il désire, par un système de "pourquoi successifs", lui permettant d'approfondir à souhait et à son choix, les réponses données par l'ordinateur.

V.2.4. Les représentants et les vendeurs

Un tel système expert peut effectivement être destiné à un représentant ou à un vendeur non spécialisé d'une grande surface. Il servirait d'outil complémentaire à la formation qu'ils subissent au début de leur service.

Ces personnes doivent être capable de proposer un système de peinture au client et de lui expliquer les principes sous-jacents. Dans le cas où un système de peinture erroné lui est soumis, il doit également pouvoir expliquer les raisons qui rejettent ce système et ceci jusqu'à un niveau de détail pouvant faire face aux désirs de tout utilisateur.

Les caractéristiques des explications à produire sont donc identiques à celles destinées aux professionnels. Il suffit de les compléter par la justification du choix d'un système de peinture.

Rappelons que les explications ne sont fournies qu'après la recherche d'un système de peinture et uniquement sur demande de l'utilisateur.

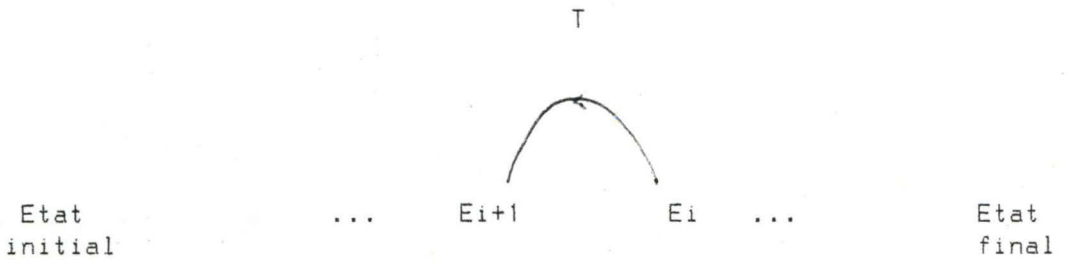
Outre le contenu, le niveau de détail et la terminologie, le sens de dérivation de la recherche d'une solution est également une caractéristique qu'il convient de prendre en considération. En effet, le système expert poursuit une recherche régressive ("backward"), puisque la solution est construite en prenant l'état final du support comme point de départ pour arriver régressivement à l'état initial du support. Cette démarche peut perturber certains utilisateurs, notamment la plupart des utilisateurs naïfs. C'est pourquoi il serait préférable d'envisager pour ce type d'utilisateur, une explication progressive ("forward") de l'état initial vers l'état final du support, bien que le système expert ait effectué une recherche régressive.

Cependant, il s'est avéré qu'une telle explication n'est pas possible à fournir ou du moins elle dépasse le cadre de notre application. Les raisons en sont les suivantes : pour fournir une explication dans le sens inverse de la recherche, plusieurs problèmes se posent.

1) Le premier problème est dû au fait que de nombreuses règles doivent être inversées.

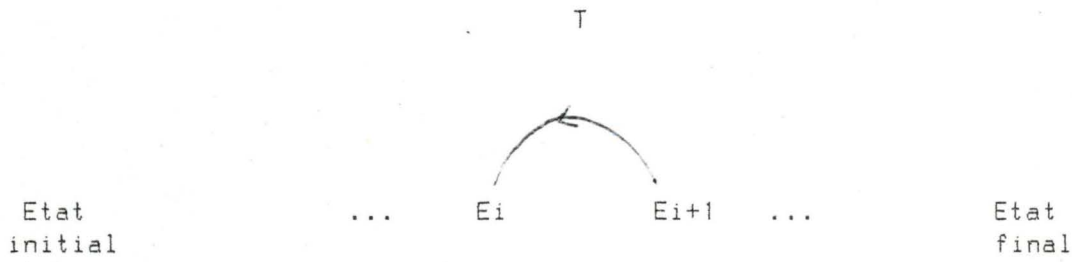
Exemple.

- Dans le sens réel de la recherche, le passage d'un état intermédiaire Ei à l'état initermédiaire suivant Ei+1 grâce au produit T se fait de la manière suivante :



De manière grossière, nous pouvons définir Ei+1 comme étant l'état Ei duquel nous avons RETIRE les caractéristiques apportées par le produit T. En effet, si par exemple, Ei comprend l'aspect "brillant" et que T est un produit par exemple donnant l'aspect "brillant", cet aspect n'est plus à satisfaire et il n'a plus à apparaître dans Ei+1.
Dans ce sens de la recherche, T est sélectionné pour satisfaire une ou plusieurs caractéristiques de Ei (il permet d'avancer vers la solution) et parce qu'il n'entre pas en contradiction avec les autres (il convient).

- Dans le sens inverse de la recherche, le passage d'un état intermédiaire Ei à l'état intermédiaire suivant Ei+1 grâce au produit T, se fait de la manière suivante :



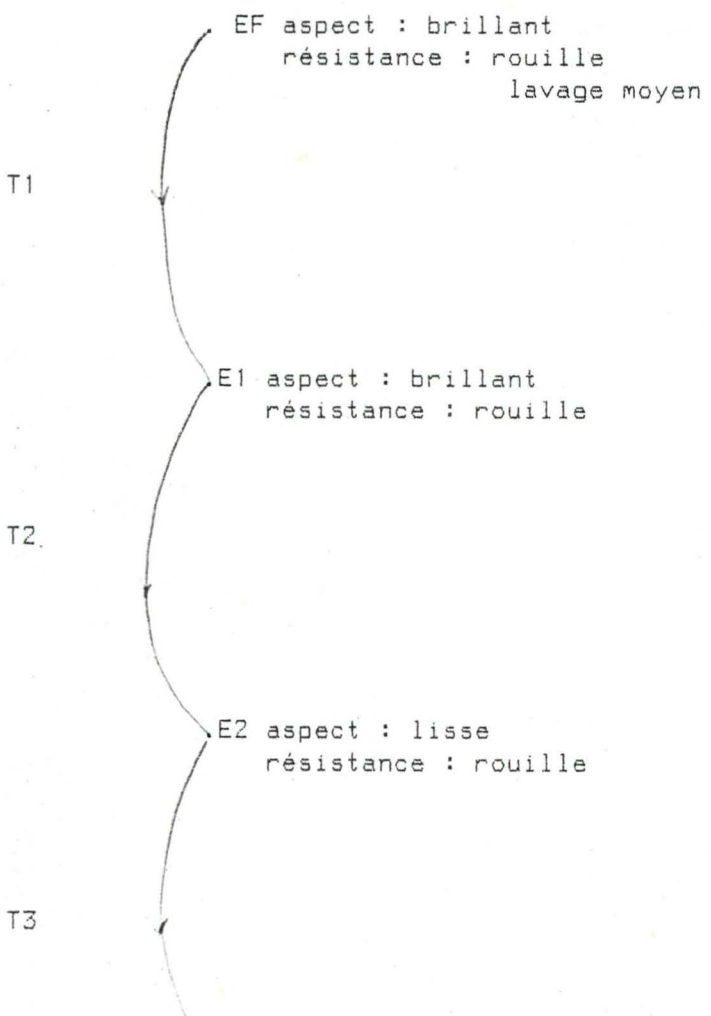
De manière grossière, nous pouvons définir Ei+1 comme étant l'état Ei auquel nous avons AJOUTE les caractéristiques apportées par le produit T. En effet, si l'état final comprend l'aspect "brillant", que cet aspect n'est pas dans Ei (cet aspect n'a pas encore été satisfait) et que T est un produit donnant l'aspect "brillant", cet aspect n'est plus à satisfaire et il doit apparaître dans Ei+1.
Dans ce sens de la recherche, T est sélectionné pour

satisfaire une ou plusieurs caractéristiques de l'état final qui n'apparaissent pas dans Ei (il permet d'avancer vers la solution) et parce qu'il n'est pas en contradiction avec les caractéristiques de Ei (il convient).
Les produits s'appliquent de manière inverse aux états.

Dans les deux sens, chaque état représente ce qui a déjà été fait. Apparemment, il suffit donc pour passer d'un sens à l'autre, d'appliquer les règles inverses de celles qui ont été réellement appliquées.
Cependant, une analyse plus fine montre qu'il existe de nouvelles règles à appliquer.

Exemple.

Soit EI, l'état initial du support et EF, l'état final du support.
Soit (T1, T2, T3), un système de peinture.



.EI aspect : lisse
résistance : /

avec T1, un traitement dont la post-condition est :
aspect : /
résistance : lavage moyen

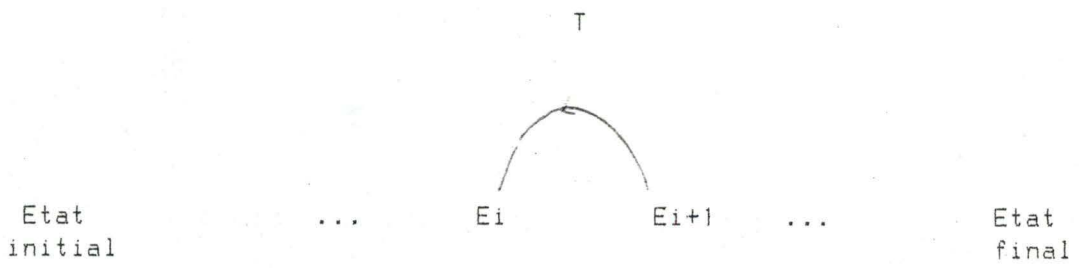
avec T2, un traitement dont la post-condition est :
aspect : brillant, lisse
résistance : /

avec T3, un traitement dont la post-condition est :
aspect : lisse
résistance : rouille.

Dans le sens réel de la recherche, T2 est un produit lisse et cet aspect étant de type "permanent", il doit nécessairement se trouver dans tous les états suivants.
Dans le sens inverse, E2 comprend "lisse" car T3 est lisse et que celui-ci est un aspect du type "permanent" (ce type correspond à la classe c) définie au point III.1), mais comment peut être expliquée sa disparition subite en E3 ?

Le passage d'un sens à l'autre implique donc l'inversion des règles réellement appliquées ET l'utilisation de règles supplémentaires (comme celles qui pourraient expliquer la disparition d'un attribut de type permanent) n'apparaissant pas du tout dans la recherche effectuée.

Des éléments supplémentaires sont également indispensables lorsque le système vérifie si le produit sélectionné n'entre pas en contradiction avec ce qui a déjà été fait dans la recherche. En effet, si le produit T permet de faire la liaison entre Ei et Ei+1, comme nous le montre ce schéma



- dans le sens réel de la recherche, le système vérifie que T convient par rapport à Ei+1
- dans le sens inverse de la recherche, par contre, le système

devrait vérifier que T convient par rapport à Ei, ce qui n'est réalisé à aucun moment lors de la recherche effective.

Or, l'explication de tout système se base sur la mémorisation de ce qui a été exécuté. L'explication d'éléments qui dépassent le champ d'application de la recherche effective ne peut se faire de cette manière. Elle nécessite une nouvelle exécution éventuellement partielle, relative uniquement à ces éléments.

2) Le deuxième problème est qu'une explication en sens inverse nécessite la distinction de deux types de produits :

- traitement visible : traitement qui est utile même dans le sens contraire, c'est-à-dire qui apporte un élément de solution qui permet de réduire la "différence" entre l'état courant et l'état but (état initial ou final du support selon le sens).

- traitement invisible : produit qui, dans le sens inverse de la recherche, n'apporte aucun élément de solution, mais qui est nécessaire au produit sélectionné précédemment dans le sens régressif de dérivation du système expert.

Exemple.
Soit le système solution ⟨T1 T2 T3⟩ où T2 est une peinture primaire. Il est possible que dans le sens régressif de recherche, T2 n'apporte rien à la solution mais T3 demandant une primaire, on ressent bien la nécessité de choisir T2. Cependant, si l'explication doit se faire de T1 vers T3, on ne ressent plus l'utilité de T2. C'est la raison pour laquelle, il faudra expliquer l'application de T1 puis l'application de T3 puis justifier le choix de T2 en fonction de T3.

Si dans le sens inverse de la recherche, le processus d'explication rencontre un traitement invisible, il devrait justifier en premier lieu le choix du prochain traitement visible, puis revenir à la justification du traitement invisible.

* Un troisième problème se pose : lorsqu'un système de peinture est rejeté par le système expert, les explications ne peuvent se faire que dans le sens de l'ordinateur car il n'est pas possible de décrire et de justifier des actions qu'il n'a pas effectuées.

Exemple.
Soit un système de peinture (T1,T2,T3,T4) proposé par l'utilisateur dans le sens état initial-état final du support. Si le système expert a envisagé, lors de sa recherche, la suite de produits (T4,T3) qui a été abandonnée en raison de la violation d'un des critères, il ne peut pas donner une justification du rejet de cette solution en commençant par les produits T1 et T2, ces traitements n'ayant pas été étudiés, du moins dans le cadre de la suite de produits (T4,T3).

Ces différentes raisons justifient le fait que le sens de dérivation n'est pas pris en considération dans notre étude. Toute explication fournie respecte le sens de la recherche, à savoir de l'état final vers l'état initial du support. Ce sens régressif est d'ailleurs le sens que les experts utilisent pour résoudre un problème de peinture. C'est donc le sens le plus naturel pour expliquer le raisonnement du système expert.

V.3. LES QUESTIONS ET LES REPONSES

Le problème abordé dans ce point, est de connaître quelles sont les questions que chaque type d'utilisateur est susceptible de poser au système expert et quelles sont les réponses que le processus d'explication doit pouvoir y apporter. Nous ne développons dans ce point que les questions relatives au problème particulier soumis par l'utilisateur.

V.3.1. Les questions

Grâce aux types d'explications vues au point V.2., nous pouvons établir la liste des questions auxquelles le système doit pouvoir faire face.

1. Le système expert propose à la fin de la recherche une suite de systèmes de peinture (cette suite peut ne contenir qu'un seul élément ou être vide) pour résoudre le problème de l'utilisateur, et celui-ci veut connaître la justification du choix d'un de ces systèmes-solution. Il s'agit donc d'une suite de produits apparaissant à l'écran.

   Un cas particulier de cette possibilité est une suite vide : l'ordinateur n'a pas trouvé de solution au problème posé. Le processus d'explication doit également pouvoir en donner une justification.

2. L'utilisateur propose un système de peinture qui n'est pas produit à l'écran. Il s'agit donc d'un système que l'utilisateur préconisait comme solution à son problème. Il souhaite savoir pourquoi le système expert a préféré une autre solution, c'est-à-dire qu'il désire connaître la justification du rejet du système qu'il propose.

3. L'utilisateur propose un produit qui ne se trouve pas dans un des systèmes-solution produits à l'écran. Dans ce cas, le processus d'explication doit être capable de donner les raisons qui ont permis d'écarter ce produit. Ce type de question est envisagé car l'utilisateur naïf n'est pas toujours capable de proposer un système complet, mais a cependant souvent, une idée du produit de base à appliquer.

4. Quand plusieurs solutions sont proposées par le système expert, elles sont classées selon un ordre de préférence. L'utilisateur peut vouloir connaître la justification de la préférence entre deux systèmes.

5. A chaque réponse que le système expert apporte à une question de l'utilisateur, celui-ci peut demander d'avoir des détails supplémentaires à propos de cette réponse ou d'un élément de cette réponse. C'est le système des "pourquoi successifs", permettant de passer à un niveau plus élevé de détail. Cependant, cette possibilité est refusée à tout utilisateur qui désire dépasser le niveau de détail qui lui est associé (cfr V.2.). En effet, des détails supplémentaires au-delà de ce niveau ne feraient qu'obscurcir l'esprit de l'utilisateur.

V.3.2. Les réponses

Le processus d'explication a besoin de connaître le type d'utilisateur pour lequel il doit fournir une explication. Comme nous l'avons vu au point V.2, cette information lui permet de déduire le type d'explication à générer c'est-à-dire les caractéristiques de son contenu, son niveau de détail et la terminologie à employer (éventuellement, la terminologie sera prise en charge par l'interface qui génère du langage naturel).

Outre ces aspects, il est nécessaire de fixer exactement ce que le système doit effectivement répondre à chaque question.

V.3.2.1. Les explications à fournir

Nous allons examiner le(s) premier(s) niveau(x) de détail des réponses à fournir pour chaque type de question. De plus amples détails sur chacun des éléments de ces premières explications peuvent être livrées si l'utilisateur le désire, grâce aux "pourquoi successifs".

1. Justification du choix d'un système-solution

Tout système-solution produit à l'écran se trouve dans l'ordre qui paraît le plus naturel à l'utilisateur, c'est-à-dire de l'état initial vers l'état final du support.

Le processus d'explication doit fournir à l'utilisateur les critères que le système a employé pour faire sa sélection et en quoi cette solution satisfait ces critères. Les explications doivent donc présenter les étapes de la recherche concernant ce système-solution. Elles comprennent * les états intermédiaires qui ont permis de faire la transition entre l'état final et l'état initial du support

* les produits qui ont permis de générer les états intermédiaires et la justification de leur sélection par rapport aux critères techniques et économiques.

Dans le cas particulier où le système Conseil n'a PAS TROUVE DE SOLUTION au problème de l'utilisateur, une justification doit également pouvoir être fournie. Celle-ci est composée de la justification du rejet de toutes les solutions candidates qui ont été envisagées lors de la recherche. Notons que si le système n'a pu trouver un système-solution, les quelques solutions analysées ont été abandonnées en raison du non respect d'un critère technique. En effet, la violation d'un critère économique n'entraine l'abandon d'une solution candidate que s'il existe d'autres solutions à poursuivre.

Une démarche intéressante aurait été de déceler la véritable raison de cette absence de solution. La raison peut être de deux natures :

- soit au moins deux exigences de l'utilisateur, dans la description du support, sont incompatibles entre elles, selon les principes fondamentaux de la peinture. Il est impossible de trouver un système de peinture qui satisfait ces deux exigences.

  Exemple : si l'utilisateur désire peindre un support situé à l'extérieur et s'il veut lui donner un aspect mat, le système Conseil ne peut lui proposer un système de peinture. En effet, ces deux exigences sont incompatibles puisqu'aucune peinture "mat" n'est pas assez résistante face à l'environnement extérieur.

- soit au moins une exigence de l'utilisateur, dans la description du support, ne peut être satisfaite par un produit de la gamme Trimetal.

Cependant, cette démarche demande une étude que nous ne pouvons réaliser dans le cadre de notre mémoire, faute de temps.

2. Justification du rejet d'un système de peinture

L'utilisateur propose une suite de produits dans l'ordre lui paraissant le plus naturel, c'est-à-dire l'ordre d'application sur le support, de l'état initial vers l'état final du support.

Le premier niveau de détail doit être constitué en bref de la raison du rejet, c'est-à-dire
* quel(s) est(sont) le(s) critère(s) qui n'ont pas été respecté(s)
* quel est le premier produit à partir de l'état final du support (recherche BACKWARD) qui ne respecte pas ce(s) critère(s).

Il s'agit donc de montrer à l'utilisateur jusqu'où le système de peinture qu'il propose est correct, et quel(s) critère(s) le produit suivant (selon l'ordre régressif de la recherche) viole. Ce produit est nommé produit fautif.

Le deuxième niveau de détail présente la justification du sous-système correct (selon la même démarche que pour la justification d'un système de peinture correct) et montre en quoi les caractéristiques du produit fautif ne respectent pas le critère violé.

3. Justification du rejet d'un produit

Le premier niveau de détail donne
* une solution candidate contenant le produit proposé par l'utilisateur
* le(s) critère(s) en fonction du(des)quel(s) cette solution candidate a été abandonnée.

Un tel produit peut se trouver dans plusieurs solutions candidates. Le système choisit celle qui satisfait le plus de critères et qui est la plus proche de l'état initial du support. Il est en effet préférable d'expliquer à l'utilisateur qu'il existe un système de peinture (si il existe effectivement) qui comprend le produit proposé et qui est proche de la solution, plutôt que de lui dire que le système qui contient ce produit est tout à fait faux techniquement.

Le deuxième niveau de détail consiste à fournir la justification du rejet de cette solution candidate selon la même démarche que celle présentée ci-dessus (2. deuxième niveau de détail).

4. Justification de la préférence entre deux systèmes-solution

Un système-solution ne peut être préférable par rapport à un autre que parce que son évaluation est moindre. Seule la fonction d'évaluation marque une distinction entre deux solutions respectant les critères techniques. Il s'agit donc de donner une justification par rapport aux critères économiques.

Le premier niveau de détail donne la valeur des critères extra-techniques pour chaque produit de chacun des deux systèmes-solution. Cela permet de montrer à l'utilisateur ce qui les différencie du point de vue économique.

Si cette différence n'est pas très grande, des détails supplémentaires sont nécessaires. Dans ce cas, sur demande de l'utilisateur, le deuxième niveau fournit
* la valeur des critères extra-techniques pour chaque solution, c'est-à-dire l'agrégation des valeurs pour chaque produit
* la valeur de fonction d'évaluation des deux systèmes-solution et les détails de son calcul

Cette démarche est également utilisée pour justifier le rejet d'une solution candidate abandonnée en raison de son manque d'intérêt économique.

V.3.2.2. La séquentialité des décisions prise par le système expert

Pour de nombreuses applications, un processus d'explication montre la séquentialité des actions effectuées par l'ordinateur et l'ensemble des choix qu'il a faits. Si nous appliquons ce principe au programme Conseil, le processus d'explication devrait fournir pour chaque étape du système de recherche toutes les solutions candidates et la justification de la poursuite ou du rejet de chacune d'elles.

Cependant, pour les types d'utilisateurs concernés par ce système expert, peu leur importe de connaître quels ont été toutes les solutions candidates envisagées et en particulier quelles sont celles qui étaient en compétition à un certain moment avec le système qu'ils proposent. Ce qui les intéresse principalement, c'est de savoir pourquoi le(s) système(s) de peinture produit(s) à l'écran a (ont) été choisi(s) et/ou pourquoi le système qu'ils préconisaient a été rejeté en faveur de celui(ceux) produit(s) à l'écran.

V.4. DIFFERENTES STRATEGIES DE MODELISATION D'UN PROCESSUS D'EXPLICATION

Dans cette section, nous allons examiner les différentes méthodes de modélisation d'un processus d'explication qui permet à un système de justifier les actions qu'il entreprend. Nous nous sommes inspirés pour ce faire de [SWA81]

Un certain nombre d'approches différentes ont été développées au cours du temps. Les principales font usage d'un texte préparé à l'avance et produisent des explications qui découlent directement du code du programme et de la trace de son exécution.

V.4.1. Explications pré-cuites

La façon la plus simple de permettre à un ordinateur de répondre à des questions à propos de ce qu'il fait, est de prévoir les questions qui pourront être posées et de stocker les réponses correspondantes, sous forme d'un texte en langue naturelle. L'ordinateur ne peut alors afficher que les textes stockés en mémoire. C'est ce que l'on appelle du texte pré-cuit, et les explications produites en produisant du texte pré-cuit sont nommées explications pré-cuites. Le type le plus simple d'explication pré-cuite sont les messages d'erreurs.

Il est relativement aisé de concevoir un petit programme fournissant les explications de ses activités en utilisant l'approche du texte pré-cuit. Dès que le programme est écrit, du texte pré-cuit

est associé à chaque partie du programme, expliquant ce que cette partie du programme fait. Quand l'utilisateur désire savoir comment l'ordinateur a atteint la solution qu'il propose, il suffit simplement d'afficher à l'écran les textes associés aux parties de programme qui ont été exécutées.

Cependant, quelques problèmes se posent.

- Toutes les questions et réponses doivent être anticipées à l'avance et le programmeur doit prévoir des réponses pour toutes les questions que l'utilisateur peut poser. Pour les grands systèmes, c'est pratiquement impossible. D'où l'incapacité du système de s'adapter à des questions spécifiques propres à un utilisateur.

  Pour le système de conseil en peinture, l'utilisateur souhaite principalement des renseignements sur la solution correspondant au problème particulier soumis à l'ordinateur. L'explication à produire dépend donc de la solution trouvée qui elle-même dépend du problème posé. Pour appliquer cette méthode, il est alors indispensable de répertorier tous les cas qui peuvent se présenter. Ceci revient à rechercher toutes les combinaisons possibles des valeurs des attributs de l'état initial et final du support. Il est clair que cette tâche est tout à fait irréalisable.

- Le fait que le code du programme et le texte qui explique ce code, peuvent être modifiés indépendamment, rend difficile d'assurer la cohérence entre ce que le programme fait et ce qu'il explique.

- Le système n'a pas de modèle conceptuel de ce qu'il explique. Une chaîne de caractères, pour l'ordinateur, est la même qu'une autre, il la traite de la même manière qu'une autre, se souciant peu de son contenu.

- Il est malaisé d'utiliser une telle approche si nous voulons produire des types d'explications assez élaborés comme des explications par analogie ou des explications à différents niveaux d'abstraction. Un récapitulatif des actions du programme n'est effectif que si le niveau de détail est choisi à l'avance. Les explications étant destinées à divers types d'utilisateurs, le système doit pouvoir s'adapter à ceux-ci en choisissant un niveau de détail approprié à chaque type.

Cette technique ne semble donc pas être indiquée pour le problème qui nous concerne. Toutefois, afin de simplifier les choses, nous nous inspirons de cette approche pour les explications indépendantes de l'exécution du programme, comme la composition d'une peinture, les grands principes de la peinture et les fiches techniques d'un traitement (explications abstraites).

Ces informations étant tout à fait indépendantes de l'application, les inconvénients de cette première approche disparaissent.

V.4.2. Exploitation de la trace et transformation du code

Dans cette approche, les explications découlent directement du programme. Les routines d'explications examinent le programme grâce à la trace (ensembles de règles qui ont été appliquées lors de l'éxécution et qui sont instanciées aux données particulières du problème traité) constituée lors de l'exécution. En effectuant des transformations relativement simples sur le code, ces routines peuvent créer des explications sur la manière dont le programme a travaillé.

Les transformations à appliquer doivent permettre de convertir une ligne de programme en du langage naturel.

Exemple.

Si le programme est écrit en Prolog, les transformations qui doivent être appliquées pourraient être les suivantes : conversion du foncteur et des paramètres selon une table de correspondance, et selon la règle, réordonnancement et/ou ajout des (groupes de) mots en vue d'exprimer la relation sous-jacente tout en respectant la syntaxe du langage naturel.

La règle "convientnat (incralac, cuivre, métaux)"
devient en langage naturel : "la nature pour laquelle la peinture incralac est adaptée, à savoir le cuivre, convient pour la nature de votre support, à savoir métaux."

A partir du moment où les routines d'explications procèdent uniquement à des transformations du code, la qualité des explications ainsi produites dépend dans une grande mesure de la façon dont le code est écrit. En particulier, la structure de base du programme n'est pas fortement altérée et les noms des variables dans les explications sont principalement les mêmes que ceux qui sont utilisés dans le programme. Si les explications doivent être compréhensibles, le système expert doit être écrit de telle sorte que sa structure soit aisée à comprendre par quiconque qui soit familier au domaine d'expertise, et les noms des variables et des procédures du programme doivent représenter des concepts qui sont significatifs pour l'utilisateur, sous peine d'avoir de nombreuses tables de correspondance.

Cette méthode présente quelques avantages.

- Si le problème et le programme sont bien structurés, les explications fournies selon cette méthode sont compréhensibles par l'utilisateur et aucune autre modification n'est à apporter. Les explications ne constituent donc pas une lourde charge pour le programmeur.
- Puisque les explications reflètent directement le code, la consistance entre explications et code est assurée.

Malgré ces avantages, il existe certains problèmes dus au couplage étroit entre code et explication.

- L'inconvénient majeur est le suivant : alors que cette technique expose ce que le système fait ou a fait, elle ne peut déterminer pourquoi le système a fait ce qu'il a fait. Le système ne peut donner les justifications de ses actions. L'information nécessaire pour justifier ses actions est l'information que le programmeur a utilisée pour écrire le programme, mais elle n'est généralement pas incorporée dans ce programme pour qu'il s'exécute correctement. A partir du moment où nous désirons que le système expert soit capable aussi bien de donner la justification de ses actions que de se dérouler avec succès, nous avons besoin de trouver un moyen de saisir les connaissances et les décisions qui sont intervenues en premier lieu dans la conception du programme, qui constituent les fondements du programme.

  Exemple : pour le conseil en peinture, les règles fondamentales (et donc celles qu'il est nécessaire d'expliquer) pourraient être les suivantes :
  1) celles qui établissent si, pour le dernier traitement sélectionné
     - sa nature convient
     - sa forme convient
     - son environnement convient
     - son aspect convient
     - son état de surface convient
     - ses caractéristiques spéciales conviennent
     - ses applications particulières conviennent
     - ses protections conviennent
     - ses résistances conviennent

     avec les caractéristiques de l'état courant
     et si deux couches superposées sont compatibles entre elles
  2) celles qui établissent la transition entre deux états intermédiaires.

- Il peut être difficile ou impossible de structurer le programme de manière telle que l'utilisateur puisse le comprendre aisément. Les principes appliqués machinalement par les experts doivent figurer explicitement dans le programme afin qu'ils soient effectivement pris en considération. Ceci oblige parfois le programmeur à écrire des opérations que l'expert fait sans même y penser. Comme toutes les opérations exécutées par l'ordinatuer seront expliquées, cela risque d'embrouiller les experts plutôt que de les éclairer.
- Il est difficile de donner un aperçu de ce qu'il s'est réellement passé. Tandis que le système est explicite en ce qui concerne la manipulation de variables, de records et de listes, il est très peu explicite sur ce que cela signifie pour le problème qui est traité. Tout est exprimé en termes de données, ce qui est très peu significatif pour l'utilisateur. Il risque en effet de ne pas saisir la correspondance avec le sujet dont il est question.
- Les explications étant la traduction de chaque instruction, elles comprennent beaucoup trop de détails d'implémentation qui importent peu à l'utilisateur et qui risquent fortement de le détourner des choses importantes.

V.4.3. Structures statique et dynamique des règles exéxutées par le système expert

L'approche développée ci-dessous est celle que nous avons adoptée pour le système de conseil en peinture, en vue de générer les explications qui dépendent directement du problème à résoudre. Nous allons dans ce point décrire les caractéristiques générales de cette méthode et nous verrons comment elle s'applique à notre programme.

Selon cette technique, la génération d'explications s'articule autour de deux structures : une structure statique et une structure dynamique.

1. Structure statique

Une structure statique est la suite des fondements du programme, la séquence des principes à appliquer pour résoudre tout problème relevant du domaine d'expertise considéré. Elle est constituée en grande partie grâce à la collaboration des experts.

Le processus d'explication doit connaître ce que contient la base de connaissances et comment elle est organisée. Il est important de savoir quels sont les éléments de la base de données et comment ces éléments interviennent dans les décisions prises lors de la recherche d'une solution, en l'occurrence la recherche d'un système de peinture correct. L'organisation de la base de données et le raisonnement à la base de la recherche influence le processus d'explication : plus l'organisation et le raisonnement sont compliqués, plus les explications sont complexes.

Dans le système Conseil, cette structure est le schéma du processus de raisonnement effectué pour rechercher un système de peinture correct et compatible aux exigences de l'utilisateur.

Cette structure détermine la suite de règles à exécuter pour obtenir un système de peinture, et l'ensemble des principes à appliquer dans le domaine de la peinture, indépendamment d'un cas particulier. Elle est limitée aux règles à expliquer. Puisque les détails de programmation et notamment le règles d'entrées/sorties ne doivent pas être expliquées à l'utilisateur, il est inutile de les représenter dans la structure statique. Les règles fondamentales contenues dans la structure statique seront déterminées ultérieurement.

Une telle structure peut être représentée par un arbre et/ou dont

- chaque noeud est le foncteur d'une règle
- tout noeud fils est un antécédent de la règle associée au noeud père.

Cet arbre correspond à la structure type de toute exécution de programme.

2. Structure dynamique.

La structure dynamique correspond à l'instanciation de la structure statique à un problème particulier. Elle contient donc les règles de la structure statique qui ont pu être appliquées pour ce problème. Une telle structure est constituée aux moment même de l'exécution. A chaque exécution du programme correspond donc une structure dynamique. Elle est généralement représentée par la trace du programme, c'est-à-dire par la suite de règles de la structure statique qui ont été appliquées, ces règles étant complétées par des paramètres instanciés selon le problème considéré. Elle peut donc avoir la forme d'une liste.

Le processus d'explication se base sur un suivi en parallèle des deux structures. Cela permet de découvrir l'endroit où il n'existe plus de correspondance entre les deux et donc de déceler la(les) règle(s) qui a(ont) échoué.

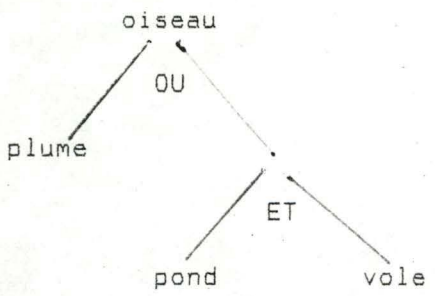
Si le système expert a trouvé une solution au problème, toutes les règles ET de la structure statique ont été exécutées avec succès et une règle de chaque sous-arbre OU a réussi. La structure dynamique contient ces règles.

Si le système expert n'a pas trouvé de solution, ou s'il a rejeté un système de peinture qui n'est pas correct, pour ce système il existe une règle qui n'a pas réussi et donc qui n'apparaît pas dans la structure dynamique .

En consultant parallèlement les deux structures, nous pouvons découvrir
- quelles sont les règles ET qui ont échoué : celles qui appartiennent à la structure statique mais qui n'appartiennent pas à la structure dynamique
- quelle est la règle qui a réussi : celle qui suit les règles OU qui on échoué.

Exemple.

Soit la structure statique suivante :



Cette structure représente les règles du programme qui détermine si un animal est un oiseau. Pour ce faire, il

doit montrer qu'il a des plumes ou qu'il pond et qu'il vole.

L'exécution du programme pour l'animal "coco" dont on sait qu'il pond donnera le résultat suivant : "coco" n'est pas un oiseau. Le système expert n'a pas assez d'informations pour pouvoir affirmer que "coco" est un oiseau. La structure dynamique pour cette exécution est la liste [pond(coco)].

En consultant ces deux structures, nous remarquons dans l'ordre en profondeur d'abord (c'est l'ordre d'exécution de l'interpréteur Prolog) que pour l'occurence "coco", la régle "plume" n'a pas réussi, que la règle "pond" a réussi et que la règle "vole" a échoué et donc que la règle oiseau a également échoué. On peut déterminer pourquoi le système n'a pu montrer que "coco" est un oiseau.

Pour plus de facilité, la structure dynamique peut contenir toutes les règles que l'interpréteur Prolog a tenté d'exécuter. Il suffit alors d'ajouter pour chaque d'elles, une variable spécifiant si elle a réussi ou échoué.

Pour le système Conseil, le contenu de ces deux structures est plus amplement détaillé dans le point V.5.3.1..

V.5. STRATEGIE D'EXPLICATION POUR LE SYSTEME CONSEIL

V.5.1. Introduction

Rappelons que nous n'envisageons dans ce chapitre que les explications dépendant directement du problème posé. Les explications complémentaires, c'est-à-dire les grands principes de la peinture, les fiches techniques d'un traitement, ...etc, font partie des extensions à apporter à notre travail.

Nous pouvons représenter le processus d'explication comme un traitement, qui étant donné certains éléments en entrée est capable de fournir des explications (cfr V.3. Les questions et les réponses).

Afin de pouvoir générer une sortie correcte en fonction de chaque entrée, le système doit se baser sur les règles qui ont été exécutées lors de la recherche. Il est donc nécessaire de mémoriser certaines informations au cours de cette recherche.

Deux possibilités nous sont offertes.

V.5.1.1. Mémorisation de la totalité des règles exécutées

Afin de pouvoir faire face à toute question, la mémorisation de la totalité des règles exécutées et le leurs paramètres, semble être nécessaire. Il suffirait alors de construire un outil de gestion permettant de sélectionner parmi toutes ces informations celles qui sont pertinentes pour répondre à la question effectivement posée.

Cependant, les systèmes de peinture concernés par les questions d'un utilisateur suite à la résolution de son problème, sont en nombre limité par rapport aux innombrables systèmes envisagés lors de la recherche. Généralement, l'utilisateur est concerné au maximum par 2 systèmes : le système de peinture qui semble être le préférable aux yeux du système expert, et le système qu'éventuellement, l'utilisateur préconisait.

De sorte que cette possibilité offre les inconvénients suivants :

- la recherche est polluée par un volume impressionnant d'informations inutiles : les informations relatives à tous les systèmes qui ne sont pas concernés par la question et les détails d'implémentation (instructions d'entrée/sortie, instructions permettant d'extraire la valeur d'un certain attribut dans un vecteur d'état, ...)

- une telle mémorisation nécessite un espace de stockage important

- cette possibilité implique également une perte de temps considérable
  * lors de la recherche; l'écriture sur fichier de chaque règle exécutée risque de ralentir très fort le système de recherche
  * lors de l'exploitation de l'ensemble des informations mémorisées; en effet, la séquentialité des informations à sélectionner ne correspond pas nécessairement à la séquentialité des informations enregistrées. Les informations nécessaires pour répondre à une question concernent une solution candidate particulière, indépendamment de tous les autres systèmes envisagés. Or, cette solution n'a pas été étudiée entièrement en une seule fois, mais le système a d'abord généré et analysé son premier produit, puis éventuellement a envisagé d'autres solutions avant de générer et d'analyser son deuxième produit, ... Il y a une discordance entre ce qui intéresse l'utilisateur (recherche en profondeur qui correspond à la construction d'UNE solution candidate) et la manière dont la recherche s'est faite (recherche heuristique). De sorte que l'information relative à un système particulier est éparpillée dans la masse d'informations mémorisées. C'est pourquoi, avant de pouvoir répondre à la question, un temps précieux doit être consacré à la recherche ardue des informations nécessaires à la formulation de cette réponse.

V.5.1.2. Réexécution locale

L'intérêt de l'utilisateur ne portant habituellement que sur deux systèmes de peinture, une réexécution locale à ces deux systèmes est toute indiquée pour pouvoir expliquer la justification de leur choix ou rejet.

Une réexécution locale à un système, consiste après la recherche initiale et après la saisie de la question, à réexécuter le processus de recherche instancié au système concerné par la question et à mémoriser les règles réexécutées. Cette réexécution correspond à une recherche en profondeur uniquement, puisque tout autre système n'est pas envisagé. Elle permet de réexécuter les règles relatives aux critères qu'une solution doit respecter et de connaître en détail ce en quoi le système concerné respecte ou viole les critères. Seules les règles appliquées pour la construction de ce système sont ainsi mémorisées.

Lorsque l'intérêt de l'utilisateur se limite à un seul produit, il convient de connaître la solution candidate la plus performante (cfr V.3.2.1.3.) dans laquelle il a été envisagé, et si la solution a échoué, le critère qui a provoqué cet échec.

Afin de répondre à ces exigences, un minimum d'informations est mémorisé pendant la recherche initiale. C'est ce que nous appelons l'historique. Une définition provisoire de ce concept est l'ensemble de toutes les suites de produits envisagées lors de la recherche.

Nous verrons que l'historique permet également de répondre selon le premier niveau de détail à toutes les questions que l'utilisateur pourrait poser. Il permet donc d'éviter une réexécution locale lorsque l'utilisateur se limite au premier niveau de détail, ce qui est le cas le plus fréquent, du moins pour l'utilisateur naïf. Nous allons voir de manière plus précise dans les chapitres suivants ce que sont exactement l'historique et la réexécution locale, comment ces mécanismes sont construits et comment ils sont exploités.

V.5.2. Historique

V.5.2.1. Définitions

Procédons à quelques définitions.

- Nature d'une suite de produits : caractéristique qualifiant la qualité de cette suite de produits par rapport aux critères techniques (d'utilité et autres) et économiques (avec ou sans question). Le premier produit de la suite est le produit de finition, c'est-à-dire le produit s'appliquant en dernier lieu sur le support.

  Une suite peut être de trois natures différentes :

* réussite par rapport à tous les critères; les solutions candidates de cette nature sont les systèmes-solution, c'est-à-dire les systèmes de peinture produits à l'écran
* échec par rapport aux critères économiques (critères extra-techniques avec et sans question) et réussite par rapport aux critères techniques
* échec par rapport aux critères économiques et techniques; pour les suites de produits de cette nature, tous les produits précédant (dans l'ordre régressif de la recherche) le produit fautif, respectent tous les critères. Le produit fautif n'est pas applicable techniquement et les produits suivants n'ont pas été envisagés puisque cette solution candidate a été abandonnée au cours de la recherche initiale.

Une suite de produits ne peut être que d'une seule nature.

La nature d'une suite de produits permet de savoir si elle a réussi ou échoué, et dans ce deuxième cas de connaître le critère qui est à la source de l'échec. Lorsqu'une suite de produits a échoué, sa nature donne la raison pour laquelle le système expert l'a abandonnée au cours de la recherche régressive.

Exemples.

-) Le système (Primer Surfacer, Permacryl) est correct techniquement, mais il n'est pas satisfaisant du point de vue économique.
-) Le système (Stellapaint) n'est pas correct techniquement car il est utile mais il ne convient pas techniquement.
-) Le système (Auber3) est correct techniquement et est satisfaisant du point de vue économique.

A l'issue de la recherche, deux cas peuvent se présenter.

1. Lorsque le système de peinture préconisé par l'utilisateur ne se trouve pas parmi les systèmes-solution, l'utilisateur désire en premier lieu connaître la raison de cet échec, c'est-à-dire sa nature.
2. Lorsque le produit préconisé par l'utilisateur ne se trouve dans aucun système-solution produit à l'écran, il désire en connaître la raison, c'est-à-dire la nature d'une suite de produits qui le comprend et qui a été envisagée pendant la recherche.

Si l'historique contient les informations rendant possible la déduction de la nature d'une suite de produits quelconque,

1. dans le premier cas, une réexécution locale peut être évitée si l'utilisateur se limite au premier niveau de détail. S'il désire de plus amples détails, la nature du sous-système permet de situer le contexte dans lequel devra se faire la réexécution locale (c'est-à-dire quelle est exactement la suite de règles à réexécuter).

2. dans le second cas, l'historique permet de voir dans quelle suite de produits le traitement préconisé a-t-il été envisagé et quelle est la nature de cette suite.

- L'historique se définit plus précisément de la manière suivante : il s'agit de l'ensemble de toutes les suites de produits envisagées lors de la recherche. Cet ensemble est subdivisé en catégories, selon la nature des suites de produits.

  Pour connaître la nature d'une suite de produits, il suffit de déterminer la catégorie dans laquelle elle se trouve, chaque catégorie étant associée à une nature.

  L'historique se présente sous forme d'une liste divisée en sous-listes pour chaque catégorie :

```
[[S11,S12,...,S1i,...],[S21,S22,...,S2j,...],...,[Sm1,Sm2,...,Smk,..]]
  \_________________/   \_________________/       \_________________/
     catégorie 1           catégorie 2              catégorie m
```

avec S11, S12, S1i, les suites de produits dont la nature est la nature associée à la catégorie 1,
S21, S22, S2j, les suites de produits dont la nature est la nature associée à la catégorie 2,
Sm1, Sm2, Smk, les suites de produits dont la nature est la nature associée à la catégorie m.

V.5.2.2. Les catégories

L'historique est donc structuré en catégories :

- la catégorie contenant les suites de produits sorties à l'écran c'est-à-dire les systèmes-solution; nous l'appellerons catégorie-réussite.

- les catégories correspondant à un échec par rapport à un critère; nous les appellerons catégories-échec.

Chaque catégorie-échec est associée à un critère. Une telle catégorie est un ensemble de suites de traitements représentant des solutions candidates qui ont été abandonnées en raison de la non satisfaction (échec) du critère associé à cette catégorie. Il est possible qu'en réalité une solution candidate viole plusieurs critères. Cependant nous ne pouvons nous baser que sur ce que le système expert a fait et donc nous considérons le premier critère violé. En effet, les autres critères n'ont pas été testés par le système puisque dès qu'il y a échec, la solution est abandonnée.

Il existe donc autant de catégories-échec que de critères. Les éléments contenus dans une catégorie-échec ne respectent pas le critère associé à cette catégorie.

Une suite de produits appartient à une seule catégorie. En effet, soit la suite de produit est sortie à l'écran, de ce fait elle appartient à la catégorie-réussite et à aucune autre. Soit la suite ne répond pas à un critère, nous l'enregistrons dès lors dans la catégorie-échec correspondant à ce critère.

* Les catégories-échec associées à un critère d'applicabilité contiennent des suites de produits dont le dernier ne satisfait pas ce critère.

* Les catégories-échec associées à un critère d'optimalité contiennent des suites de produits applicables mais qui ne respectent pas ce critère.
  Les catégories-échec associées à un critère d'optimalité peuvent être affinées. On les divisent en deux parties :

  - une catégorie contenant les suites de produits qui ne respectent pas les critères et qui forment une solution candidate complète

  - une catégorie contenant les suites de produits qui ne respectent pas les critères et qui forment une solution partielle.

  Toute catégorie-échec peut donc être remplacée par deux catégories-échec : l'une contenant les solutions complètes et l'autre contenant les solutions partielles.

  Cela nous permet de connaître une information supplémentaire. En effet, lorsqu'il y a échec pour un critère d'optimalité, la solution rejetée peut être partielle ou complète. La subdivision en deux parties permet de connaître quelles sont les suites de produits qui constituent une solution au problème posé d'un point de vue applicabilité, si les critères d'optimalité ne sont pas pris en considération.

  Une solution candidate échouant par rapport à un critère d'applicabilité ne peut être complète par définition. Pour une catégorie associée à un tel critère, il n'y a pas de distinction à faire.

GENERALISATION

De manière générale, toute recherche se fait selon un nombre quelconque m de critères. Ces critères ne pouvant être pris en considération exactement au même moment, le processus de recherche établit un ordre selon lequel les critères sont envisagés. Nous avons vu précédemment qu'il existait des critères d'applicabilité et des critères d'optimalité et habituellement les critères d'applicabilité sont envisagés dans la recherche avant les critères d'optimalité.

Soient m critères : crt1
crt2
.
.
crtm

avec crti, le ième critère considéré lors de la recherche d'une solution au problème posé,

crtj, (1 <= j < n) et (1 < n <= m) sont des critères d'applicabilité

crti, (n <= i <= m) et (1 < n <= m) sont des critères d'optimalité.

En vue d'enrichir toute solution partielle Sp, le processus de recherche tente d'ajouter un produit P en vérifiant les critères selon l'ordre 1, 2, ... m. Dès qu'un critère i est violé, la solution Sp est abandonnée bien qu'elle satisfait les critères 1, 2, ... i-1. La suite de produits (Sp, P) appartient à la catégorie i, associée au critère i.

La catégorie Ci peut donc se définir de la manière suivante : Ci est l'ensemble des suites de produits satisfaisant les critères 1,2

Les différentes catégories peuvent donc être ordonnées de la manière suivante, en commençant par les catégories contenant des systèmes satisfaisant le plus de critères :

- catégorie comprenant l'ensemble des systèmes-solution (solutions produites à l'écran)
- catégorie comprenant les solutions complètes échouant à cause du critère m
- catégorie comprenant les solutions complètes échouant à cause du critère m-1

- ...

- catégorie comprenant les solutions complètes échouant à cause du critère n

- catégorie comprenant l'ensemble des solutions non complètes échouant à cause du critère m

- catégorie comprenant l'ensemble des solutions non complètes échouant à cause du critère m-1

- ...

- catégorie comprenant l'ensemble des solutions non complètes échouant à cause du critère n

- catégorie comprenant l'ensemble des solutions non complètes échouant à cause du critère n-1

- ...

- catégorie comprenant l'ensemble des solutions non complètes échouant à cause du critère 1.

Selon cet ordre, une suite de produits appartenant à une catégorie satisfait les objectifs associés aux critères des catégories suivantes.

APPLICATION AU SYSTEME CONSEIL

Les critères pris en considération par le processus de recherche sont au nombre de 3 :
- critère économique (critère d'optimalité)
- critère d'applicabilité technique \
- critère d'utilité / critères d'applicabilité.

Les différentes catégories formant l'historique sont dès lors les suivantes :

C1 --> ensemble des solutions produites à l'écran (ensemble des systèmes-solution)

C2 --> ensemble des solutions complètes ne satisfaisant pas le critère économique

C3 --> ensemble des solutions partielles ne satisfaisant pas le critère économique

C4 --> ensemble des solutions partielles ne satisfaisant pas le critère technique

C5 --> ensemble des solutions partielles auxquelles le système a tenté d'ajouter un traitement qui n'apporte rien au problème, qui n'est pas utile.

La catégorie constituée des solutions complètes ne satisfaisant pas le critère technique est vide puisque par définition, une solution complète respecte le critère technique. Cette catégorie n'est donc pas à considérer.

Selon la répertorisation des catégories définie au cours de la généralisation, la dernière catégorie devrait contenir toutes les suites de produits dont le dernier produit ne satisfait pas le premier critère et donc aucun des critères. Cette catégorie risque d'être très volumineuse puisque généralement il existe beaucoup plus de produits non applicables qu'applicables. De sorte que nous avons

décidé de limiter le contenu de cette catégorie aux suites de produits pour lesquelles on a tenté d'ajouter un produit qui ne satisfait pas ce premier critère. Nous verrons l'utilité de cette catégorie lors de l'exploitation de l'historique (cfr V.5.2.4.).

V.5.2.3. Constitution de l'historique

Le contenu des catégories constituant l'historique étant fixé, comment procéder progressivement à sa création lors de la recherche d'une solution au problème de l'utilisateur ? C'est ce dont il est question dans ce point.

Au cours de la recherche, le système expert doit mémoriser

* chaque suite de produits abandonnée dans la catégorie correspondant au critère responsable de cet échec
* les systèmes-solution dans la catégorie-réussite.

Par conséquent, une suite de produits n'est à mémoriser qu'à la fin de sa construction, si c'est un système-solution. Par contre, si c'est un système partiel, c'est à la découverte du critère violé qu'il doit être mémorisé.

Lorsque le système expert étudie un produit en vue de l'ajouter à la solution partielle déjà obtenue, plusieurs cas peuvent se présenter :

- soit le produit satisfait tous les critères et la nouvelle suite de produits ainsi formée est une solution complète. La nouvelle solution est mémorisée dans la catégorie C1 contenant les systèmes-solution.

- soit le produit satisfait tous les critères et la nouvelle suite de produits ainsi formée n'est pas complète. Cette suite de produits n'est pas abandonnée à ce stade de la recherche. Elle sera complétée ultérieurement. Le système expert continue sa recherche sans devoir mémoriser quoi que ce soit à ce niveau.

- soit le produit satisfait le critère qui est en train d'être considéré. Le système peut considérer le critère suivant sans devoir mémoriser quoi que ce soit à ce niveau.

- soit le produit ne satisfait pas le critère qui est en train d'être considéré. La suite de produits initiale complétée de ce produit est mémorisée dans la catégorie associée à ce critère.
  S'il s'agit du critère économique, la nouvelle suite de produits est mémorisée dans l'une des deux catégories correspondant à ce critère selon qu'elle constitue un système partiel ou complet (C3 ou C2).
  S'il s'agit du critère technique (autre que le critère d'utilité), la nouvelle suite de produits est mémorisée dans la catégorie correspondant à ce critère, C4.
  S'il s'agit du premier critère (le critère d'utilité), seule la suite initiale est mémorisée dans la catégorie C5.

V.5.2.4. Exploitation de l'historique

Quand la recherche est terminée, l'utilisateur peut poser certaines questions. Ces questions se rapportent toujours à une suite de produits ou à un seul produit. Le premier niveau de détail de la réponse, consiste à découvrir la nature de cette suite de produits ou de celle du produit préconisé par l'utilisateur. Le processus d'explication doit donc tenter de voir à quelle catégorie cette suite appartient.

La consultation de l'historique se fait des catégories les plus restrictives vers les catégories les moins restrictives. Les plus restrictives sont celles qui contiennent des suites de produits qui satisfont le plus de critères. Cet ordre trouve sa justification dans le fait que les catégories plus restrictives contiennent beaucoup moins d'éléments, et que dans le cas où l'utilisateur propose un seul produit, la suite de produits qui le contient doit être celle qui satisfait le plus de critères.

Nous vous proposons dans ce qui suit deux types d'utilisations de l'historique : la première pour connaître la justification d'une suite de produits, la seconde pour connaître la justification d'un produit. Il s'agit d'une justification du premier niveau de détail c'est-à-dire de la nature.

A. L'utilisateur propose une suite de produits dont il désire la justification

Cette suite de produits est donnée dans le sens le plus naturel pour l'utilisateur, c'est-à-dire de l'état initial à l'état final du support. Or, la recherche sur ordinateur s'est faite dans le sens inverse, de sorte que chaque élément de toute catégorie est également dans ce sens.

La première opération à effectuer par le processus d'explication est donc d'inverser le système soumis par l'utilisateur. Dès lors, un système-utilisateur se définit par l'inverse de la suite de produits proposée par l'utilisateur.

Comme nous l'avons vu précédemment, le premier niveau de détail d'explication à fournir pour un système de peinture, est de donner la nature de ce système grâce à une recherche dans l'historique.

Nous abordons ci-dessous la manière de faire la recherche dans l'historique et l'explication du premier niveau à fournir pour un système-utilisateur. Un algorithme de recherche sera fourni. Nous verrons ensuite quelle a été la formalisation choisie.

Soient S = (T4, T3, T2, T1) une suite de produits proposée par l'utilisateur qui en désire une justification
et S'= (T1, T2, T3, T4), le système-utilisateur.

Afin de connaître la nature de S c'est-à-dire pour savoir si S est un système-solution ou si c'est une solution qui a été abandonnée en cours de recherche, il faut procéder à une consultation de l'historique.

Dès que l'on connait la nature de S' et donc de S, il sera possible de donner une justification détaillée du choix ou du rejet de ce système. C'est l'explication du premier niveau. Si l'utilisateur désire de plus amples détails, il pourra accéder au niveau 2 d'explication (cfr V.5.3.).

- soit S' $\in$ C1 => S est une solution qui a été produite à l'écran (système-solution)

- soit S' $\in$ C2 => S est une solution complète mais ne satisfait pas le critère économique

- soit S' $\in$ C3 => S est une solution non complète et elle ne satisfait pas le critère économique

- soit S' $\in$ C4 => S est une solution non complète et T4 n'est pas applicable aux autres traitements

- soit S' $\in$ C5 => S n'est pas une solution complète.

- Avant de descendre d'une catégorie, il convient de vérifier si S' est préfixe d'un élément d'une catégorie. Dans ce cas, S est un système non complet.
  Ce cas nous intéresse car grâce à un produit que l'on ajoute, on peut présenter à l'utilisateur un système supérieur à S.

Exemple. Le système (T1, T2, T3, T4, T5) $\in$ C2 : S' est un système non complet mais le processus d'explication signale à l'utilisateur que S a été envisagé dans le système (T1, T2, T3, T4, T5) qui est un système correct techniquement bien qu'il soit peu satisfaisant du point de vue économique.

Ce cas n'est pas équivalent à S' appartenant à C5. Car on ne peut ajouter un de produit (car aucun n'est techniquement applicable) aux suites de produits de C5. Par conséquent, S ne peut être amélioré.

Si aucun de ces cas ne s'est présenté, on envisage le système S1 = (T1, T2, T3).

Plusieurs cas peuvent se présenter à la fin de la recherche dans l'historique :

- soit S1 $\in$ C1 => S comprend trop de traitements. S1 est une solution correcte

- soit S1 $\in$ C2 => S comprend trop de traitements. S1 est une solution complète mais elle ne satisfait pas le critère économique

- soit S1 $\in$ C3 => S1 ne satisfait pas le critère économique

- soit S1 $\in$ C4 => le dernier traitement de S1 n'est pas applicable par rapport aux autres

- soit S1 $\in$ C5 => (T1 T2 T3) est une suite de produits applicables => le problème se situe au niveau de T4. T4 ne peut pas être non applicable techniquement puisque S n'appartient pas à C4. T4 est donc un produit non utile.

- Avant de descendre d'une catégorie, il convient de vérifier si S1 est préfixe d'un élément d'une catégorie. Dans ce cas, S1 n'est pas un système complet, car on peut lui rajouter un traitement qui lui donne la nature de la catégorie étudiée.

Si aucun de ces cas ne s'est présenté, on recommence le même processus en supprimant T3 puis T2. Si de nouveau aucun de ces cas ne se présente, cela signifie que T1 ne satisfait aucun des critères.

Grâce à cette recherche, nous connaissons la nature de S, c'est-à-dire si c'est un système-solution ou si c'est une suite de produits qui a échoué et dans ce cas le critère à la source de cet échec.

Afin d'expliquer le cas particulier où il n'y a pas de système-solution à l'issue de la recherche, le processus d'explication expose dans un premier temps (premier niveau de détail), le contenu des catégories C4 et C5 et la nature de chacun de leurs systèmes. Rappelons que dans ce cas spécifique, les catégories C1, C2 et C3 sont vides puisque qu'on ne peut abandonner une solution pour des raisons économiques s'il n'existe pas d'autre solution à proposer à l'utilisateur.

Algorithme de recherche

- Situation initiale

Après la saisie du système de peinture pour lequel l'utilisateur désire une justification, le système-utilisateur est calculé. Il est nommé S.

* Soit S appartient à la catégorie contenant les systèmes solution
  => S est un système-solution.

* Soit S appartient à une catégorie associée à un critère d'optimalité
  => S est un système ne satisfaisant pas ce critère.

* Soit S appartient à une catégorie associée à un critère d'applicabilité
  => S n'est pas un système complet et le dernier produit de S ne satisfait pas ce critère.

* Soit S appartient à la dernière catégorie
  => S n'est pas un système complet et il n'est pas possible de le compléter pour former une solution terminale.

* Soit S est préfixe d'un élément d'une catégorie quelconque
  => S est un système non complet.

Si aucun de ces cas ne s'est présenté, il faut envisager le système S1 = S \ {trtn} avec trtn le dernier traitement de S.

- Situation générale

Le système expert dispose d'une suite de traitements S1 dont il doit faire la recherche dans l'historique.

* Soit S1 appartient à une des catégories autre que la dernière
  => S comprend trop de traitements et S1 est un système défini selon la catégorie.

* Soit S1 appartient à la dernière catégorie
  => S1 est une suite de produits applicables. Le problème se situe au niveau de trtn qui n'est pas applicable ni même utile.

* Soit S1 est préfixe d'une catégorie
  => S1 n'est pas un système complet car on peut lui rajouter un (des) traitement(s) qui lui donne(nt) la nature de la catégorie étudiée.

Si aucune condition d'arrêt ne s'est présentée, S1 est mis à jour en lui ôtant le dernier traitement.

- Situation finale

Plusieurs cas peuvent provoquer l'arrêt de la recherche :

* un des sous-systèmes analysés appartient (en tout ou en parite) à une catégorie. L'explication correspondant à cette catégorie est produite à l'écran.

* auncun des sous-systèmes analysés n'appartient (en tout ou en partie) à une catégorie et S1 est une suite vide. Cela signifie que le premier traitement de S n'est pas applicable techniquement. Cette explication est produite à l'écran.

Formalisation

La formalisation que nous avons choisie pour cet algorithme, se base sur deux définitions :

- système-historique : suite de produits appartenant à une catégorie de l'historique et correspondant au système-utilisateur

- système-rendu : suite de produits inclue dans un système-utilisateur et ayant permis de définir exactement la nature du système-utilisateur. Il s'agit donc de la partie du système-utilisateur envisagée lors de la dernière étape du mécanisme.

Si le système-utilisateur est identique au système-rendu (notation : système-utilisateur != système-rendu), la nature du système-utilisateur a été trouvée, dès le premier cycle. Un système tel que S1 n'a pas dû être envisagé.

Si le système-utilisateur n'est pas identique au système-rendu, c'est-à-dire que le système-rendu est le système-utilisateur duquel certains produits ont été supprimés (notation : système-utilisateur >! système-rendu), un seul cycle n'a pas suffi pour mener à bien la recherche à travers les catégories.

Quatre cas peuvent se présenter et en fonction de chacun d'eux, un type d'explication est à fournir.

1) système-utilisateur != système-rendu != système-historique :
le système-utilisateur se trouve intégralement dans une catégorie et l'explication à fournir est la nature correspondant à cette catégorie.

2) système-utilisateur != système-rendu et
système-rendu !< système-historique :
le système-utilisateur n'est pas complet. Il est un sous-système du système-historique dont la nature est celle correspondant à la catégorie à laquelle il appartient.

3) système-utilisateur >! système-rendu et
système-rendu != système-historique :
le système-historique est contenu dans le système-utilisateur et sa nature est celle correspondant à la catégorie à laquelle il appartient.

4) système-utilisateur >! système-rendu et
système-rendu !< système-historique : la première partie du système-utilisateur, c'est-à-dire le système-rendu est un sous-système du système-historique, dont la nature est celle correspondant à la catégorie à laquelle il appartient.

EXEMPLE DE CHACUN DE CES CAS

A l'issue de la recherche d'une solution à un problème particulier, l'historique est le suivant :

C1 --> [Silva+gloss].
[Stellor].
[Permacryl, Multiprimer].

C2 --> [Permacryl, Primersurfacer].
[Permacryl, Stelprimer].
[Permacryl, Auber3].
[Tpcyacht].
[Aubexol].

C4 --> [Auber3].
[Primersurfacer].
[Stellapaint].
[Stelprimer].

C5 --> [Permacryl].

C3 est vide.

1) Soit [Permacryl, Primersurfacer], le système-utilisateur.
Le système expert fournit à l'utilisateur l'explication suivante :
"Ce système est correct techniquement mais est trop peu satisfaisant du point de vue économique."
Le système-rendu est [Permacryl, Primersurfacer] et le système-historique est [Permacryl, Primersurfacer].

2) Soit [Permacryl] le système-utilisateur.
Le système expert fournit à l'utilisateur l'explication suivante :
"Ce système a trop peu de traitements.
[Permacryl, Multiprimer] est plus complet. Ce système est correct techniquement et est satisfaisant du point de vue économique."
Le système-rendu est [Permacryl] et le système-historique est [Permacryl, Multiprimer].

3) Soit [Stellor, Multiprimer], le système-utilisateur.
Le système expert fournit à l'utilisateur l'explication suivante :
"Ce système contient trop de traitements. Il contient le système [Stellor]. Ce système est correct techniquement et est satisfaisant du point de vue économique."
Le système-rendu est [Stellor] et le système-historique est [Stellor].

4) Soit [Permacryl, Incralac], le système-utilisateur.
Le système expert fournit à l'utilisateur l'explication suivante :
"Ce système n'est pas correct techniquement. Le système [Permacryl, Multiprimer] est correct techniquement et est peu satisfaisant du point de vue économique."
Le système-rendu est [Permacryl] et le système-historique est [Permacryl, Multiprimer].

B. Soit T, le traitement proposé par l'utilisateur qui en désire une justification

Etant donné le traitement T proposé par l'utilisateur, plusieurs cas peuvent se présenter :

- soit T est contenu dans un élément de C1
  => T appartient à une solution produite à l'écran

- soit T est contenu dans un élément de C2
  => T appartient à une solution qui n'est pas acceptée économiquement

- soit T est contenu dans un élément de C3
  => T appartient à un système qui est abandonné en cours de recherche pour une raison économique

- soit T est contenu dans un élément de C4
  => T appartient à un système qui est abandonné en cours de recherche pour une raison technique

- soit T est contenu dans un élément de C5
  => T appartient à un système qui n'admet plus de produits utiles ou applicables, sinon il aurait été possible de trouver T dans une des catégories précédentes

- soit T n'est contenu dans aucune catégorie
  => T n'est pas utile et ne satisfait aucun des critères.

Algorithme de recherche

Afin de fournir une explication (de premier niveau) cohérente sur le traitement T, il faut faire la recherche suivante :

- soit T appartient à un élément de la catégorie des systèmes-solution
  => T appartient à un système de peinture correct techniquement et satisfaisant du point de vue économique

- soit T appartient à un élément d'une des catégories (autre que la catégorie contenant les systèmes-solution)
  => T appartient à un système dont l'échec est dû au critère associé à la catégorie

- soit T n'appartient pas à un élément d'une catégorie
  => T ne satisfait aucun des critères.

Bien entendu, lors de la phase d'explication, les notions d'historique et de catégories sont transparentes à l'utilisateur.

V.5.3. Réexécution locale

Dans ce qui suit, nous exposons l'utilité de faire une réexécution locale. Ensuite, nous expliquons tout d'abord, ce que contient la trace locale et la manière dont celle-ci est constituée et exploitée. Et finalement, en quelques mots, nous décrivons les problèmes de la génération de phrases explicatives.

Etant donné la nature d'un système de peinture intéressant l'utilisateur, si celui-ci désire de plus amples détails sur la construction de ce système, une réexécution locale s'avère nécessaire.

a) Justification du système solution

La justification développe les critères d'applicabilité, en l'occurence techniques, selon lesquels ce système a été construit.

Une bonne justification technique est, à partir de l'énoncé du problème, de retrouver tous les états intermédiaires entre l'état final et l'état initial et d'expliquer le passage entre ces états grâce aux produits.

La raison commerciale du choix d'un système de peinture est tout simplement le fait qu'il est le plus intéressant du point de vue commercial, aussi bien pour l'utilisateur que pour la firme. Les deux niveaux de la justification commerciale sont décrits à la section V.3.2.1. au point 4.

b) Justification du rejet d'un système de peinture

- La suite de produits correspondante trouvée dans une catégorie est techniquement correcte mais est moins intéressante du point de vue commercial. Dans ce cas, le processus d'explication génère une justification technique (cfr. (a)) pour cette suite de produits et signale que celle-ci n'est pas très commerciale comparativement au(x) système(s)-solution.

- La suite de produits correspondante trouvée dans une catégorie révèle l'échec technique d'un produit. Le processus d'explication génère alors une justification technique pour les produits précédant le produit qui provoque l'échec, et une justification de l'échec technique de ce traitement. Le processus d'explication

fournit une justification d'applicabilité jusqu'au produit fautif.

Ces justifications constituent le niveau de détail supérieur au premier niveau (celui-ci décrivant la nature d'un système). Une explication de plus en plus précise doit pouvoir être fournie sur demande de l'utilisateur, grâce au mécanisme des "pourquoi successifs" permettant de descendre progressivement dans les niveaux de détail.

Des explications adéquates aux désirs de l'utilisateur, sont fournies grâce à l'exploitation de l'ensemble des informations mémorisées pendant la réexécution locale au système de peinture concerné. Cet ensemble est nommé trace locale.

V.5.3.1. Contenu de la trace locale

Idéalement, pour avoir le plus d'informations possible sur ce que l'ordinateur a fait , la trace devrait contenir toutes les instructions (et leurs paramètres) qui ont été exécutées pour un problème particulier. Il suffirait alors de construire un outil de gestion afin d'exploiter au mieux ces informations et d'en extraire celles qui nous intéressent.

Cependant, la trace dans sa totalité est inutile puisqu'elle comprend des détails d'implémentation et des éléments techniques dépassant le champ d'intérêt de l'utilisateur. Seuls les éléments nécessaires à la production d'un type quelconque d'explication doivent impérativement être stockés.

Il suffit de mémoriser les règles qui permettent d'expliquer le raisonnement suivi dans le choix ou le rejet d'un système de peinture. Pour un système de peinture déterminé, la trace locale doit donc permettre de construire les états intermédiaires Si et d'expliquer la transition Ti entre chacun d'eux, selon le niveau de détail désiré par l'utilisateur.

Dans un premier temps, nous allons classer les règles dans le but de générer des explications selon différents niveaux de détail. Ensuite, nous expliquons la manière dont nous avons formalisé la trace locale.

a) Classification des règles et formalisation des niveaux de détail

La classification des règles permet de raisonner à partir de quelles règles il est possible de générer les explications souhaitées et de donner une formalisation au niveau de détail.

Les règles se subdivisent en deux classes principales :

(1) les règles pour lesquelles aucune explication n'est produite

Exemple : - règles d'entrée /sortie comme la saisie des données du problème et l'affichage du (des) systèmes-solution
- règles de calcul et d'extraction de données : il est inutile d'expliquer comment un attribut a été ôté de son vecteur d'état pour en connaître la valeur.

(2) les règles statégiques pour lesquelles le processus d'explication doit pouvoir fournir des informations. Chacune de ces règles est associée à un niveau de détail qui exprime son degré de généralité. Les règles les plus générales, les moins explicites, ont un niveau peu élevé. Les règles contenant beaucoup de détails, ont un niveau très élevé.

On peut donc affirmer la chose suivante : les antécédents d'une règle ont un niveau qui est égal au niveau de cette règle, augmenté de un.

Toute règle est associée à un seul niveau qui représente son niveau de détail et tout type d'utilisateur est associé à un niveau de détail maximum pour lequel le processus d'explication ne peut développer les règles dont le niveau lui est supérieur.

L'ensemble de ces règles peut être schématisé par un arbre où

* un noeud représente le foncteur d'une règle
* un arc entre deux noeuds représente le fait que le noeud fils est un antécédent du noeud père.

Le niveau d'une règle se définit par la profondeur de son conséquent.

Le premier et le dernier niveau de cet arbre sont à fixer en fonction de ce qui a été décidé au point V.2..

* Le premier niveau
  Le premier niveau ne découle pas nécessairement du programme. Il est fixé par le concepteur du processus d'explication. Il doit au moins contenir une règle correspondant à chaque critère. En effet, la réussite ou l'échec d'un critère doit au moins pouvoir être expliquée de manière générale.

  - Les règles correspondant à chaque critère d'applicabilité (en l'occurrence le critère technique et le critère d'utilité) servent à expliquer le choix ou le rejet d'un produit appartenant à une

solution candidate dont il faut expliquer la construction.

- Les règles correspondant à chaque critère d'optimalité (en l'occurrence le critère économique) servent à expliquer l'adoption ou le rejet une suite de produits.

* Le deuxième niveau
Le deuxième niveau est constitué des enfants, des antécédents des règles du premier niveau, si ces antécédents existent et s'ils n'appartiennent pas à l'ensemble des règles pour lesquelles aucune explication n'est produite.

..
..

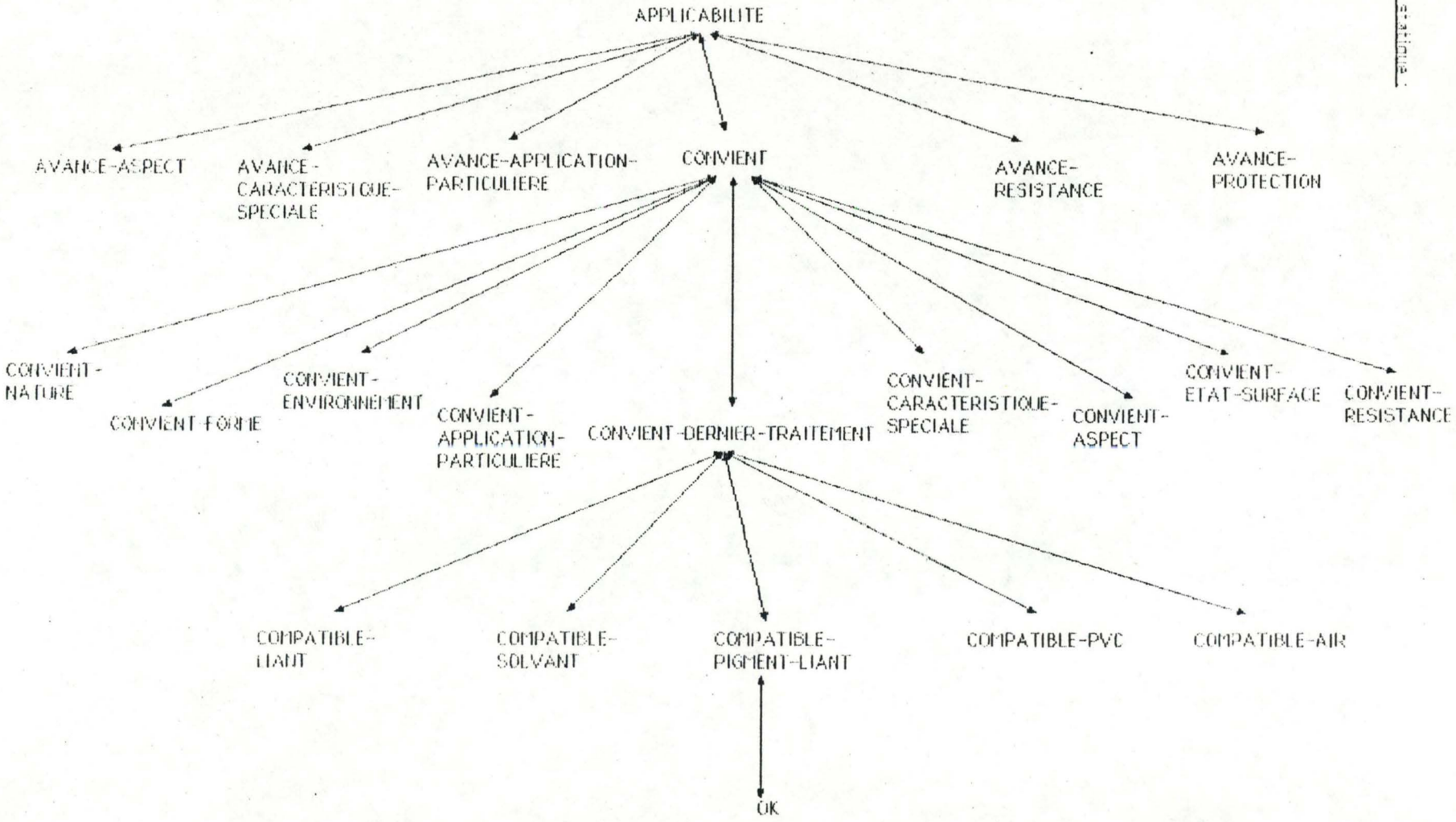
* Le ième niveau
Le ième niveau est constitué des enfants, des antécédents des règles du (i-1)ième niveau, si ces sous-buts existent et s'ils n'appartiennent pas à l'ensemble des règles pour lesquelles aucune explication n'est produite.

..
..

* Le dernier niveau
Le dernier niveau est également défini par le concepteur et représente le niveau en-dessous duquel aucun type d'utilisateur n'a accès. Il est constitué des feuilles de l'arbre. Il s'agit des règles qui n'ont besoin d'aucune autre règle pour être expliquée. Ce sont des règles pour lesquelles, il n'est pas possible ou pas nécessaire de donner plus de détails, quel que soit le type de l'utilisateur. Ce niveau comprend notamment les règles de manipulation de données où il est préférable de ne pas fournir les détails.

Rappelons que le premier niveau de cet arbre constitue en fait le deuxième niveau d'une explication, puisque le premier est la nature du système ou du produit proposé par l'utilisateur.

Cette arbre constitue la structure statique.


APPLICABILITE
AVANCE-ASPECT
AVANCE-CARACTERISTIQUE-SPECIALE
AVANCE-APPLICATION-PARTICULIERE
CONVIENT
AVANCE-RESISTANCE
AVANCE-PROTECTION
CONVIENT-NATURE
CONVIENT-FORME
CONVIENT-ENVIRONNEMENT
CONVIENT-APPLICATION-PARTICULIERE
CONVIENT-DERNIER-TRAITEMENT
CONVIENT-CARACTERISTIQUE-SPECIALE
CONVIENT-ASPECT
CONVIENT-ETAT-SURFACE
CONVIENT-RESISTANCE
COMPATIBLE-LIANT
COMPATIBLE-SOLVANT
COMPATIBLE-PIGMENT-LIANT
COMPATIBLE-PVC
COMPATIBLE-AIR
OK

Remarquons que la structure statique ne contient que des règles relatives aux critères techniques. En effet, la justification économique d'un système de peinture est constituée

* des caractéristiques extra-techniques des produits qui composent ce système. Il est très aisé d'extraire ces catactéristiques de la base de données au moment de fournir l'explication.

* de la fonction d'évaluation de ce système de peinture. Cette évaluation est le résultat d'un calcul que le système expert veillera à mémoriser lors de la réexécution locale.

La justification économique prend également en considération les critères extra-techniques avec question. De sorte qu'il est primordial de mémoriser les questions posées à l'utilisateur lors de la recherche initiale et les réponses qu'il y a apportées. Cela permet de lui montrer comment ses préférences sont entrées en ligne de compte dans le choix ou le rejet d'un système.

FORMALISATION DE LA STRUCTURE STATIQUE

Pour toute règle ayant des antécédents (ou "enfants"), le fait suivant fait partie de la structure statique :

enfant (FONCTEUR, LISTEENFANTS)

avec FONCTEUR, le foncteur de la règle (nom de la règle)
LISTEENFANTS, la liste des foncteurs des enfants de la règle.

Pour toute règle ayant un parent, le fait suivant fait partie de la structure statique :

parent (FONCTEUR, LISTEPARENT)

avec FONCTEUR, le foncteur de la règle (nom de la règle)
LISTEPARENT, la liste du foncteur du parent de la règle.

EXEMPLE

```
enfant (convient, [convientna,     convientforme,    convientenv,
                   convientap,     convientcarspec,   convienta,
                   convientes,      convientres,      convientpro,
                   convientdt ]).

parent (convientnat, [convient]).
parent (convientforme, [convient]).
parent (convientenv, [convient]).
parent (convientap, [convient]).
parent (convientcarspec, [convient]).
parent (convienta, [convient]).
```

```
parent (convientes, [convient]).
parent (convientres, [convient]).
parent (convientpro, [convient]).
parent (convientdt, [convient]).
```

Pour toute règle de la structure statique, il est nécessaire de connaître :

- la liste de ses enfants, pour pouvoir expliquer ceux-ci, en cas de "pourquoi successifs"

- son parent, pour pouvoir expliquer celui-ci, en cas de "comment successifs".

Lors de la réexécution locale, les règles réellement exécutées PARMI CELLES DE LA STRUCTURE STATIQUE, doivent être mémorisées. C'est ce que nous appelons la structure dynamique ou trace locale.

Pour chacune de ces règles, le processus d'explication doit également disposer d'un certain nombre d'informations :

- son foncteur, afin de connaître exactement quelle règle a été exécutée

- ses paramètres instanciés, afin de faire le lien avec le problème posé

- son signe : il faut savoir si la règle a réussi ou a échoué afin de savoir s'il faut expliquer sa réussite ou son échec

- ses données globales : les données ne faisant pas partie des paramètres de la règle ou des paramètres de ses enfants, mais des données relatives au contexte.

  Exemple.
  La règle "convient-aspect" ne comprend pas le traitement dont il est question. Il est donc nécessaire de mémoriser le nom de ce traitement pour expliquer à l'utilisateur que c'est la nature de celui-ci qui convient.

- ses données locales : les données ne faisant pas partie des paramètres de la règle, mais bien des paramètres de ses enfants et qui sont nécessaires pour fournir une explication cohérente.

  Exemple.
  Foncteur : convient-aspect.
  Signe : réussi.
  Paramètres instanciés : brillant, filmogène (aspect "permanent" et "à remplacer" de l'état courant (cfr III.1)).
  Donnée globale instanciée : Permacryl (traitement considéré).
  Données locales instanciées : coloré, opaque, brillant, filmogène, lisse (les aspects donnés par le traitement).

Grâce à ces informations, la règle s'explique comme suit :
"Le traitement Permacryl donne les aspects [coloré, opaque, brillant, filmogène, lisse]. Il convient puisqu'il couvre les aspects "permanents" et "à remplacer" qui sont compris dans les aspects [brillant, filmogène] spécifiés dans l'état."

Le mécanisme de dérivation de l'explication à partir d'une règle sera développé à la section V.5.3.4.

Si l'utilisateur demande une explication supplémentaire sur une règle n'appartenant pas au dernier niveau, plusieurs cas sont à envisager :

- ses enfants sont dans un sous-arbre ET et la règle a réussi : il convient d'expliquer chacun de ses enfants

- ses enfants sont dans un sous-arbre ET et la règle a échoué : il convient d'expliquer le premier enfant qui a échoué

- ses enfants sont dans un sous-arbre OU et la règle a réussi : il convient d'expliquer le premier enfant qui a réussi

- ses enfants sont dans un sous-arbre OU et la règle a échoué : il convient d'expliquer l'échec de tous les enfants.

Le fait qu'il s'agit d'un sous-arbre ET ou d'un sous-arbre OU peut être déduit lors de l'exploitation de la trace locale (cfr V.5.3.3.).

Expliquer un enfant consiste en fait à générer une phrase en langue naturelle grâce à son foncteur, son signe, ses paramètres et ses données globales et locales instanciés au problème particulier (la trace locale se rapporte toujours à un problème spécifique). Cette phrase permet de globaliser les idées contenues dans la réussite ou l'échec de la règle correspondant à cet enfant.

Pour les règles n'ayant pas d'enfant, à savoir les feuilles de la structure statique, l'utilisateur ne peut avoir accès au mécanisme des "pourquoi successifs". Le problème d'expliquer leurs enfants ne se posent donc pas. Le système explique à l'utilisateur qu'il ne peut poser cette question, en raison des détails très techniques et donc peu intéressants qu'elle comporte.

Lorsqu'il existe plusieurs règles de même foncteur et que l'une d'elle a été exécutée, la trace locale contient entre autre chose, son foncteur. Cependant, le foncteur à lui seul n'est pas suffisant puisqu'il ne permet pas de déduire exactement quelle est la règle exécutée.

Plusieurs optiques sont possibles.

- L'instruction SPY : dans tout système Prolog, il existe une instruction prédéfinie SPY (FONCTEUR), qui est utilisée lorsque nous voulons focaliser notre attention sur les règles de foncteur FONCTEUR. Ces règles avec leurs paramètres et enfants, sont

produites à l'écran au fur et à mesure de leur exécution.
Pour notre application, il suffirait de changer le flux de sortie, c'est-à-dire que ce qui est produit à l'écran serait mémorisé dans un fichier, afin de cacher ces opérations à l'utilisateur tout en ayant la possibilité de les analyser. En effet, la mémorisation des enfants permettrait de savoir quelle est la règle qui a été réellement exécutée.
Outre le fait que cette possibilité présente l'inconvénient de demander une certaine recherche pour savoir quelle est la règle qui a été réellement exécutée, un changement du flux de sortie de l'instruction SPY, exige une modification dans l'interpréteur PROLOG.

- Identification d'une règle par un numéro : chaque règle de la structure statique est complétée par un numéro qui rend possible la distinction de règles différentes de même foncteur.
Bien que l'introduction de tels numéros, exige une modification du programme, c'est l'optique que nous avons choisie en raison de sa facilité.

Aux informations nécessaires à l'explication d'une règle, s'ajoute nécessairement le numéro de cette règle.

b) Formalisation de la trace locale

La trace locale a la forme suivante :

TRT1.
ETAT1.
LISTER.
CHTRT.
TRT2.
ETAT2.
.
.
.

avec TRT1, le premier traitement qui a été analysé,
ETAT1, l'état intermédiaire à partir duquel le système a tenté d'appliquer le traitement TRT1,
LISTER, la liste des règles de la structure statique qui ont été exécutées relativement au traitement TRT1 et de leurs informations,
CHTRT, une suite de caractères spéciaux qui indique le changement de traitement analysé,
TRT2, le traitement suivant qui a été analysé,
ETAT2, l'état intermédiaire à partir duquel le système a tenté d'appliquer le traitement TRT2. C'est aussi l'état obtenu après application de TRT1.

Les traitements TRT1, TRT2, ... sont les traitements du système de peinture à expliquer, précédant le traitement qui a provoqué le rejet

de ce système, s'il a été abandonné en cours de recherche. S'il s'agit d'un système adopté par le système, la liste des traitements TRT1, TRT2, ... sont tous les traitements de ce système.

Dans LISTER, les règles exécutées et appartenant à la structure statique sont mémorisées sous la forme suivante :

[FONCTEUR , N , LISTED]

avec FONCTEUR, le foncteur de la règle,
N, le numéro de la règle si celle-ci a réussi
la mention "échec", si la règle a échoué,
LISTED, la liste des paramètres et des données globales et locales nécessaires à l'explication de la règle. Il est inutile de faire la distinction entre les différentes natures de ces éléments puisqu'ils subissent le même traitement : ils interviennent au même titre dans la phrase d'explication apparaissant à l'écran et relative à la règle.

Exemples de formalisation de règle :

-) [convient-aspect, 2, [Permacryl, [brillant, filmogène], [coloré, opaque, brillant, filmogène, lisse]]].

La deuxième règle de foncteur "convient-aspect" a réussi. Cela signifie que "Le traitement Permacryl, donne les aspects [coloré, opaque, brillant, filmogène, lisse]. Il convient puisqu'il couvre les aspects permanents et à remplacer qui sont compris dans les aspects [brillant, filmogène] spécifiés dans l'état.".

-) [convient-dernier-traitement,échec,[Primersurfacer, Permacryl]].

La règle "convient-dernier-traitement" a échoué. Cela signifie par exemple que "Le traitement Primersurfacer ne convient pas car les liants de Permacryl ne sont pas compatibles avec les liants du traitement Primersurfacer".

Le mécanisme de dérivation de l'explication à partir d'une règle sera développé à la section V.5.3.4.

V.5.3.2. Constitution de la trace locale

La réexécution locale à un système de peinture doit, pour être fructueuse, s'accompagner de la mémorisation de la trace locale.

Pour ce faire, toute règle qui appartient à la structure statique et qui a été exécutée (avec réussite ou non) doit être mémorisée dans un fichier. Pour ce faire, des modifications du programme doivent malheureusement être apportées. Ces modifications ne doivent

évidemment rien changer à la logique de la procédure. Ce ne sont que les effets de bord qui varient.

Naturellement, il faut lors de la recherche des systèmes ne pas mémoriser ces règles. Par conséquent, il y a deux modes d'emplois pour les règles appartenant à la structure statique, l'un pour la recherche générale, l'autre pour la trace locale.

Nous utilisons pour résoudre ce problème un indicateur.

imprime(1) : cet indicateur est un prédicat PROLOG. S'il n'est pas dans la base de faits, alors les règles ne doivent rien mémoriser. Par contre, s'il est dans la base de faits, il faut mémoriser les informations.

Pour une bonne utilisation de cet indicateur, lors de la constitution de la trace locale, le systeme devra écrire dans la base de faits "imprime(1)". Lors de la recherche des solutions, le système devra enlever, s'il y est, le fait "imprime(1)" de sa base de faits.

Les règles devront simplement vérifier si l'indicateur est dans la base de faits. S'il n'y est pas, rien n'est à mémoriser sinon le système écrit les règles exécutées sur un fichier.

Les modifications à apporter au programme sont décrites ci-dessous.

- chaque règle de la structure statique doit avoir la forme suivante :

```
FONCTEUR(LISTEPAR) :- corps de la règle,
                      (imprime(1),
                       write( [ FONCTEUR , N , LISTED ] );
                       true).
```

avec imprime (1), indicateur,
FONCTEUR, le foncteur de la règle,
N, le numéro de la règle,
LISTEPAR, la liste des paramètres,
LISTED, la liste des paramètres et des données globales et locales nécessaires à l'explication de la règle,
true, permet de rendre la règle vraie au cas où imprimme (1) est faux, c'est-à-dire si le processus se trouve dans la recherche initiale

Si le corps de la règle a réussi, la règle vérifie
si "imprime(1)" est dans la base de fait
alors on écrit sur le fichier et on a fini,
sinon on passe par true et on a fini.

- chaque procédure appartenant à la structure statique doit être complétée d'une nouvelle règle, en fin de cette procédure.

En effet, si on veut mémoriser l'échec de la procédure, on doit écrire une règle qui échoue toujours. Elle aura comme fonction : si

le prédicat "imprime(1)" est dans la base de faits, elle mémorise sur un fichier l'échec de la procédure. Ensuite cette règle échoue.

```
FONCTEUR(LISTEPAR) :- imprime(1),
                      write([ FONCTEUR , échec, LISTED ] )
                      false.
```

avec LISTEPAR, la liste des paramètres,
imprime (1), indicateur,
FONCTEUR, le foncteur de la règle,
LISTED, la liste des paramètres et des données globales et locales nécessaires à l'explication de la règle,
false, fait échouer la règle.

Si "imprime(1)" n'est pas dans la base de données la règle échoue,
sinon l'échec de la procédure est mémorisé sur le fichier. Ensuite, on l'a fait échouer grâce à false.

Quelques problèmes de programmation sont exposés à la section VI.2.5.1.2.

V.5.3.3. Exploitation de la trace locale

La trace locale est utile en un premier temps, pour donner des explications sur un système de peinture, ensuite pour expliquer une règle plus en détail. Nous finirons ce paragraphe en vous donnant un exemple.

1. L'explication d'une suite de produit

Etant donné une suite de produits et la trace locale à cette suite, comment le sytème expert peut-il en expliquer la construction et son choix ou son rejet ?

Pour tout traitement TRT contenu dans la trace locale (structure dynamique), le système expert

* recherche dans la structure statique les règles de premier niveau

* consulte la structure dynamique afin de voir si ces règles ont réussi (grâce à son foncteur et au deuxième membre de sa représentation - crf V.5.3.1.b.)

* si tel est le cas, le traitement TRT a été adopté techniquement et l'état intermédiaire résultant de l'application de TRT mémorisé dans la trace locale, est produit à l'écran.
  Si une de ces règles n'a pas réussi, le système expert signale

l'échec de cette règle à l'utilisateur (échec technique de la solution candidate).
Si toutes les règles de premier niveau ont réussi pour le traitement étudié, et si la trace contient un traitement suivant, la solution formée par les traitements déjà analysés a été acceptée économiquement. Sinon, le dernier traitement étudié a provoqué l'abandon de la solution candidate pour des raisons économiques. Le système signale à l'utilisateur cet échec économique.

Pour le système Conseil, les règles de premier niveau sont avance-aspect, avance-application-particulière, avance-etat, avance-résistance, avance-protection et convient.

Les 5 premières règles sont celles qui testent si le traitement est utile. Elles forment un sous-arbre OU. En effet, un traitement est utile s'il fait avancer la recherche grâce à son application particulière OU à son aspect OU à son état de surface OU à ses résistances OU à ses protections.

La règle "convient" vérifie si le traitement convient au support sur lequel on l'applique et au traitement précédent.

2. L'explication d'une règle concernant un produit

Si l'utilisateur, après avoir eu l'explication de son système de peinture, désire de plus amples détails sur un des points de cette explication, Le système expert, grâce à son mécanisme de "Pourquoi et Comment successifs", va développer la règle relative à ce point obscur.

La règle à expliquer est définie par la question de l'utilisateur :

- "Pourquoi N Trt ?" : c'est la règle d'identifiant N à laquelle le traitement Trt se rapporte.
- "Comment N Trt ?" : c'est le parent de la règle d'identifiant N à laquelle le traitement Trt se rapporte.

Le système expert recherche ses enfants dans la structure statique. Le fait que ces enfants sont dans un sous-arbre ET ou OU n'y est pas spécifié mais il peut être déduit grâce à la trace locale. En effet,

- si la règle a réussi et si dans la trace locale
  * tous les enfants ont réussi, alors les enfants forment un sous-arbre ET
  * des enfants ont échoué, un enfant a réussi et les autres sont absents, alors les enfants forment un sous-arbre OU

- si la règle a échoué et si dans la trace locale
  * des enfants ont réussi, un enfant a échoué et les autres sont absents, alors les enfants forment un sous-arbre ET
  * tous les enfants ont échoué, alors les enfants forment un sous-arbre OU.

Le fait que la règle à expliquer a réussi ou échoué, est une information contenue dans la trace locale.

Le fait de savoir si les enfants forment un sous-arbre ET ou OU est nécessaire afin de savoir comment relier les explications concernant chaque enfant pour ainsi obtenir un texte cohérent.

Se pose alors un problème pour les règles du premier niveau qui n'ont pas de parents. La démarche que nous avons adoptée pour savoir quelles sont les règles dans un sous-arbre ET ou dans un sous-arbre OU, est de donner un parent fictif aux règles de sous-arbre OU.

Ces règles sont regroupées dans la règle faitavancer dont le rôle est fictif.

```
faitavancer :- avance-aspect;
               avance-application-particulière;
               avance-etat;
               avance-résistance;
               avance-protection;
```

3. Exemple

Le type d'explication fournie par le système Conseil pour un système-solution (T1, T2, T3) (dans l'ordre inverse : de l'état final à l'état initial) et un état final "brillant, anti-rouille, lavage moyen", est le suivant.

S.E. (système expert) :

- T1 convient en fonction de
  - * sa nature [Règle 1]
  - * son environnement [Règle 2]
  - * sa forme [Règle 3]
  - * son aspect [Règle 4]
  - * ses résistances [Règle 5]
  - * ses protections [Règle 6]
  - * ses applications particulières [Règle 7]
  - * son état de surface [Règle 8]
  - * ses caractéristiques spéciales [Règle 9].

  T1 fait avancer pour son aspect "brillant". [Règle 11]

- L'état devient "anti-rouille, lavage moyen". [Règle 12]

- T2 convient en fonction de
  - * sa nature [Règle 1]
  - * son environnement [Règle 2]
  - * sa forme [Règle 3]
  - * son aspect [Règle 4]
  - * ses résistances [Règle 5]
  - * ses protections [Règle 6]

* ses applications particulières [Règle 7]
* son état de surface [Règle 8]
* ses caractéristiques spéciales [Règle 9]

et est compatible avec T1 [Règle 10]

T2 fait avancer pour sa résistance "lavagege fort". [Règle 11]

- L'état devient "anti-rouille". [Règle 12].

- T3 ne fait pas avancer. [Règle 11].

U. (utilisateur) : pourquoi T1 1 ? (l'utilisateur désire de plus amples renseignements sur la règle 1 appliquée pour le traitement T1)

S.E. : la nature de T1 qui est "bois" convient pour le support qui est de nature "bois". [Règle 13]

U. : comment T1 13 ? (l'utilisateur désire remonter un niveau au-dessus de la règle 13)

S.E. : T1 convient en fonction de sa nature. [Règle 1]

U. : pourquoi T2 11 ?

S.E. : T2 procure une résistance contre le lavage fort et donc contre le lavage moyen qui est moins exigeant. [Règle 14]

V.5.3.4. Modélisation des règles et génération de phrases explicatives

Le but de ce point est de trouver un modèle qui permettrait à partir d'une règle et des informations nécessaires pour l'expliquer (signe, paramètres, données globales et locales) contenues dans la trace locale, de générer du langage naturel globalisant l'idée sous-jacente à la réussite ou l'échec de la règle.

Exemple. La représentation de la règle "convient" dans la trace locale

[convient, 1, [permacryl]],

doit être modélisée de manière à ce que le système puisse générer sans trop de problèmes, la phrase suivante :

"Le traitement permacryl convient en fonction de sa nature [Règle 1], de son environnement [Règle 2], de sa forme [Règle 3], de son aspect [Règle 4], de ses résistances [Règle 5], de ses protections [Règle 6], de ses applications particulières [Règle 7], de son état de surface [Règle 8], de ses caractéristiques spéciales [Règle 9]. Le traitement permacryl est compatible avec T1 [Règle 10]."

Une idée serait de classer les règles selon leur similitude (les règles travaillant sur des objets différents, mais ayant la même fonction). Dans notre application, les procédures "convient-nature, convient-environnement" ont une fonction identique : vérifier que le traitement convient sur le support, l'une en fonction de sa nature, l'autre en fonction de son environnement. A partir de ces classes, les phrases à générer, auraient toutes la même structure.

Exemples.

- Classe Cl1 : convient-nature, convient-environnement, ...
- Classe Cl2 : avance-aspect, avance-protection, ...

Pour les règles de la classe Cl1,nous aurions la phrase "Le traitement est approprié pour A1 de A2, Il convient donc pour A1 A3 spécifiés dans l'état." A1, A2, A3 sont des paramètres à instancier. Ainsi si nous devons expliquer la règle suivante : convient-environnement(urbain, rural) : il faudrait instancier A1 à "l'environnement", A2 à urbain et A3 à rural. Ce qui donnerait la phrase : " Le traitement est approprié pour l'environnement urbain. Il convient donc pour l'environnement rural spécifié dans l'état."

Cependant, le temps nous faisant défaut, une telle étude n'a pu être réalisée. Il s'agit donc d'une extension possible à notre mémoire.

La méthode simplifiée que nous avons adoptée consiste à associer à chaque règle une phrase expliquant cette règle avec son signe, ses paramètres et ses données globales et locales. Ensuite nous recherchons ses enfants grâce à la structure statique, nous recherchons ceux qui ont le même signe que la règle (réussi ou échec) dans la structure dynamique et nous les présentons. Des phrases pré-cuites sont stockées sous forme de faits. Il n'y a donc aucun système de génération automatique.

VI. CONSTRUCTION DE CONSEIL

Ce chapitre explique dans un premier temps la structure du programme Conseil. Ensuite, le lecteur découvre les problèmes de l'implémentation : la description des types d'objets et les spécifications des procédures du programme.

VI.1. ELABORATION DE LA STRUCTURE DE CONSEIL

Dans cette section, nous voyons l'évolution du système Conseil et de sa structure : ses premiers pas, avec la recherche systématique de toutes les solutions possibles, ensuite l'ajout des critères commerciaux, et en dernier lieu sa capacité d'expliquer son raisonnement.

VI.1.1. La recherche systématique des solutions

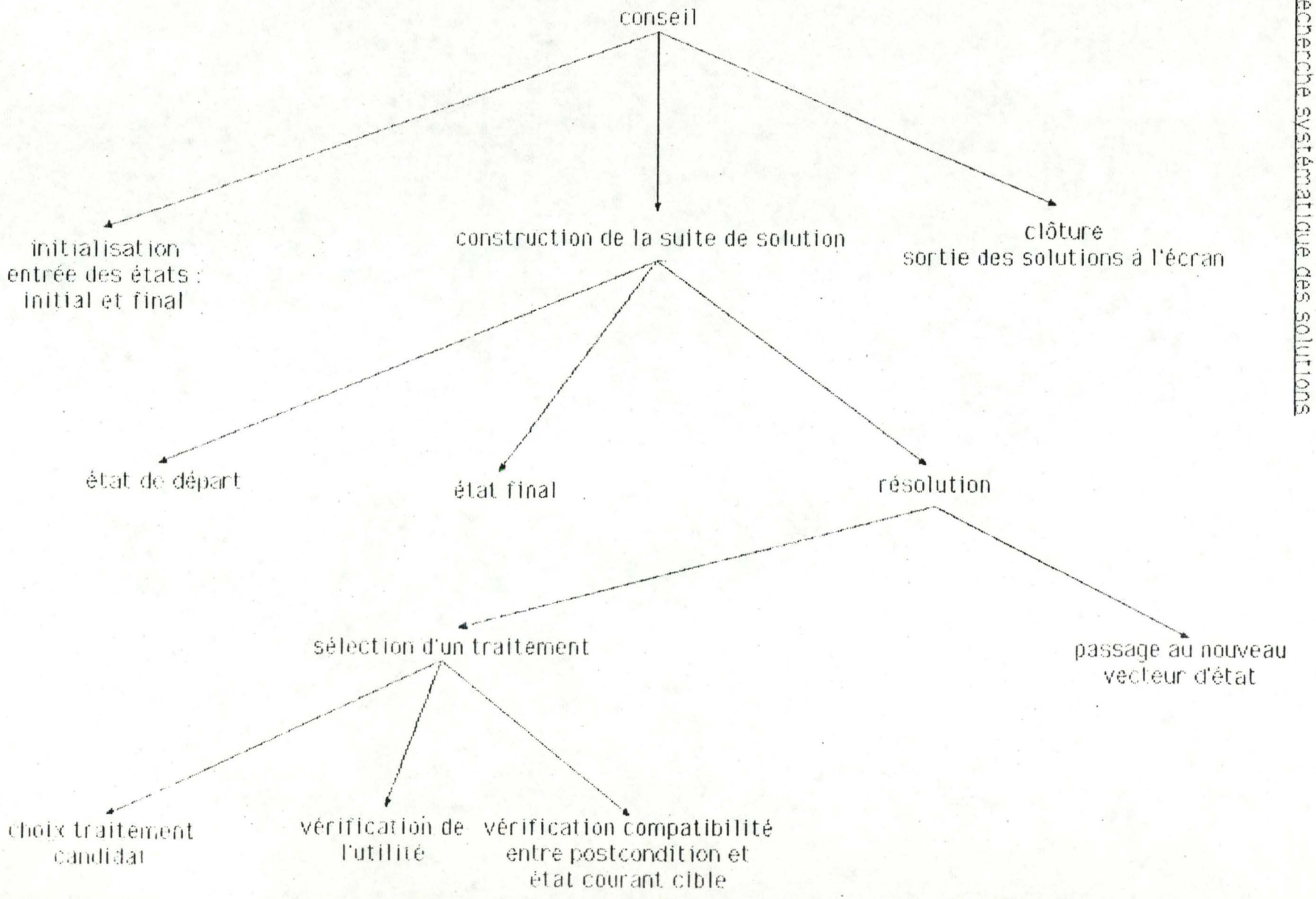
Dans sa première phase, le système Conseil est destiné à résoudre un seul type de problème : à partir de l'état initial et de l'état final d'un support à peindre, donner les suites de traitements que le support dans son état initial doit subir pour atteindre son état final. Cette suite respecte les critères techniques.

Le système Conseil effectue 3 tâches :

- l'introduction des données
- la recherche selon les critères techniques
- la sortie à l'écran des solutions.

Ainsi un premier arbre "et" des tâches, peut s'établir comme schématisé ci-après. Cet arbre est une étude détaillée du processus de construction de la séquence résultat sous forme d'une approche "top-down".

Cette phase de Conseil a été réalisée dans [DEC86].


conseil
initialisation
entrée des états :
initial et final
construction de la suite de solution
clôture
sortie des solutions à l'écran
état de départ
état final
résolution
sélection d'un traitement
passage au nouveau
vecteur d'état
choix traitement
candidat
vérification de
l'utilité
vérification compatibilité
entre postcondition et
état courant cible

VI.1.2. Prise en compte des critères commerciaux

Grâce à l'ajout des critères commerciaux, le système Conseil a pour fonction supplémentaire de donner la suite de traitements la plus intéressante pour l'utilisateur.

1° Les liens avec la première phase du système Conseil

Les concepts de base de la première phase sont les suivants :

- ETAT DU SUPPORT : est défini comme l'état d'un objet susceptible de subir un traitement
- TRAITEMENT : est l'application d'un produit de la gamme Trimetal.

Ces concepts sont expliqués dans [DEC86] dans la section 4.2.2.

Après avoir discuté avec des experts sur le problème du choix de la meilleure séquence de traitements à appliquer sur le support, il nous est apparu qu'il manquait des caractéristiques aux concepts de base.

TRAITEMENT : un TRAITEMENT n'a pas seulement des caractéristiques techniques mais aussi des caractéristiques plus commerciales. Ces caractéristiques sont détaillées dans la section IV.2.2.

ETAT DU SUPPORT : l'ETAT DU SUPPORT a également des caractéristiques commerciales. Ces caractéristiques économiques sont les mêmes que celles d'un TRAITEMENT.

Exemple. Pour l'état final, le prix au mètre carré est nul. En effet, on n'a encore appliqué aucun traitement. Pour l'état initial, le prix au mètre carré est égal au prix des traitements que l'on a appliqués sur ce support pour atteindre cet état.

A ces caractéristiques s'ajoutent deux autres :

- solution : les traitements qui ont été appliqués sur cet état.
- évaluation de l'ETAT DU SUPPORT : valeur de la fonction d'évaluation calculée à partir de l'ETAT DU SUPPORT.

2° Le processus de raisonnement

Le système Conseil doit générer la ou les meilleures solution(s) possible(s) d'une part, et d'autre part, il doit le faire de façon optimale.

La première tâche du système Conseil est l'introduction d'une description de l'état initial, de l'état final et du nombre de solutions désirées par l'utilisateur.

La deuxième tâche est la construction de la ou les solution(s). Cette tâche se divise en sous-tâches :

a) la sélection d'une solution partielle

Ayant plusieurs solutions partielles (c'est à dire des ETATs DU SUPPORT qui ne sont pas l'état initial désiré), nous choisissons celle dont la valeur de la fonction d'évaluation est la plus petite, en d'autres termes celle qui a le plus de chances d'être la solution la plus intéressante.

N.B. : au début de la recherche, la solution partielle choisie est la solution unique représentant l'état final du support.

b) la résolution

Ayant une solution partielle, nous choisissons le traitement qui convient et qui fait progresser la solution vers l'état initial.

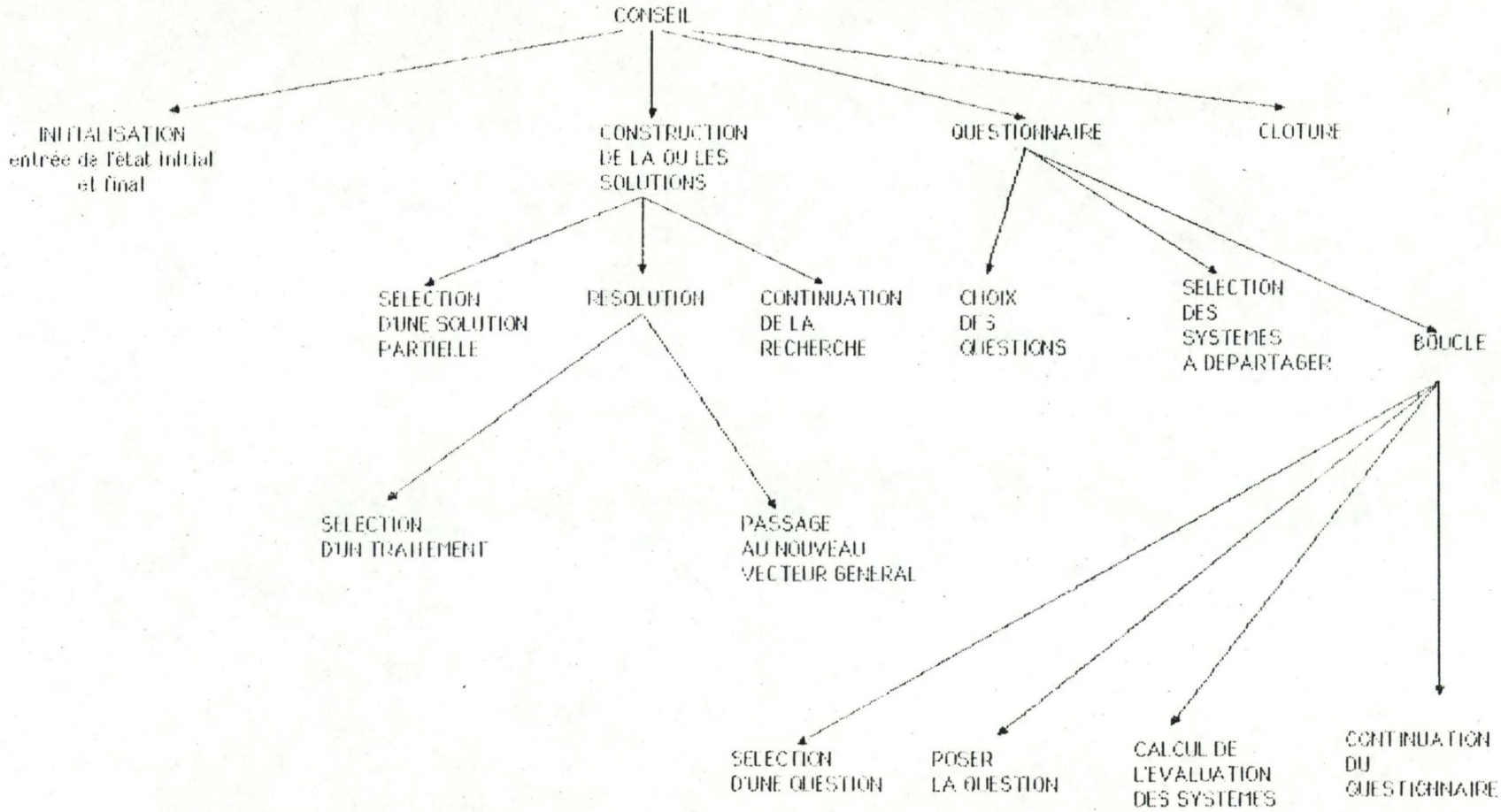
c) la continuation de la recherche

Nous vérifions dans cette partie, si avec cette nouvelle solution, il est nécessaire de continuer la recherche.

La troisième tâche est le questionnaire : ayant un nombre plus important de solutions terminales désirées, le système Conseil pose des questions à l'utilisateur, afin de différencier les solutions terminales. Cette tâche est présentée à la section IV.4.2.2.2.

La quatrième tâche est la sortie à l'écran (clôture) des solutions terminales qui ont été trouvées grâce à la deuxième et troisième tâches.

Ainsi, un premier arbre "et" des tâches peut s'établir comme schématisé à la page suivante. Cet arbre est une étude détaillée du processus de raisonnement du système Conseil prenant en considération les critères commerciaux.


CONSEIL
INITIALISATION
entrée de l'état initial
et final
CONSTRUCTION
DE LA OU LES
SOLUTIONS
QUESTIONNAIRE
CLOTURE
SELECTION
D'UNE SOLUTION
PARTIELLE
RESOLUTION
CONTINUATION
DE LA
RECHERCHE
CHOIX
DES
QUESTIONS
SELECTION
DES
SYSTEMES
A DEPARTAGER
BOUCLE
SELECTION
D'UN TRAITEMENT
PASSAGE
AU NOUVEAU
VECTEUR GENERAL
SELECTION
D'UNE QUESTION
POSER
LA QUESTION
CALCUL DE
L'EVALUATION
DES SYSTEMES
CONTINUATION
DU
QUESTIONNAIRE

VI.1.3. Le processus d'explication

Les experts, après avoir donné une solution pour un problème, sont capables d'expliquer le raisonnement qu'ils ont suivi. Il doit en être de même avec les systèmes experts.

Après avoir discuté avec des chimistes, des commerciaux, des clients, nous avons défini différents types et différents niveaux d'explications. Chaque type d'utilisateur a un vocabulaire propre, a besoin de plus ou de moins de détails.

Dans ce qui suit, nous présentons la structure du raisonnement du processus d'explication.

1°) Les explications découlant de l'historique

Le système Conseil, lors de la recherche de solutions, constitue l'historique (défini à la section V.5.2).

Avec cet ensemble d'informations, le système Conseil peut donner des explications sur :

a) la recherche de l'utilité d'un traitement, c'est-à-dire sortir à l'écran la solution où le traitement est utilisé.

b) l'explication de l'inexistence de solution, c'est-à-dire sortir à l'écran les catégories 4 et 5.

c) la nature d'une solution ; cette tâche est divisée en deux sous-tâches :

- la recherche historique qui a comme objectif de trouver dans une catégorie, le système se rapprochant le plus possible du système donné par l'utilisateur
- l'identification de la nature qui a comme objectif de déterminer la nature du système donné par l'utilisateur. Avec cette identification du système, nous pouvons sortir à l'écran, les phrases correspondantes.

2°) Les explications découlant de la trace locale

Si l'utilisateur n'est pas satisfait des explications données à l'aide de l'historique, le système Conseil, comme un expert, va se remémorer certains points de sa recherche. C'est la trace locale. Celle-ci est obtenue en exécutant certaines règles dont on a instancié des paramètres (cfr section V.5.3).

Avec cette information, le système donne des explications générales sur la solution ou des explications commerciales d'une solution finale.

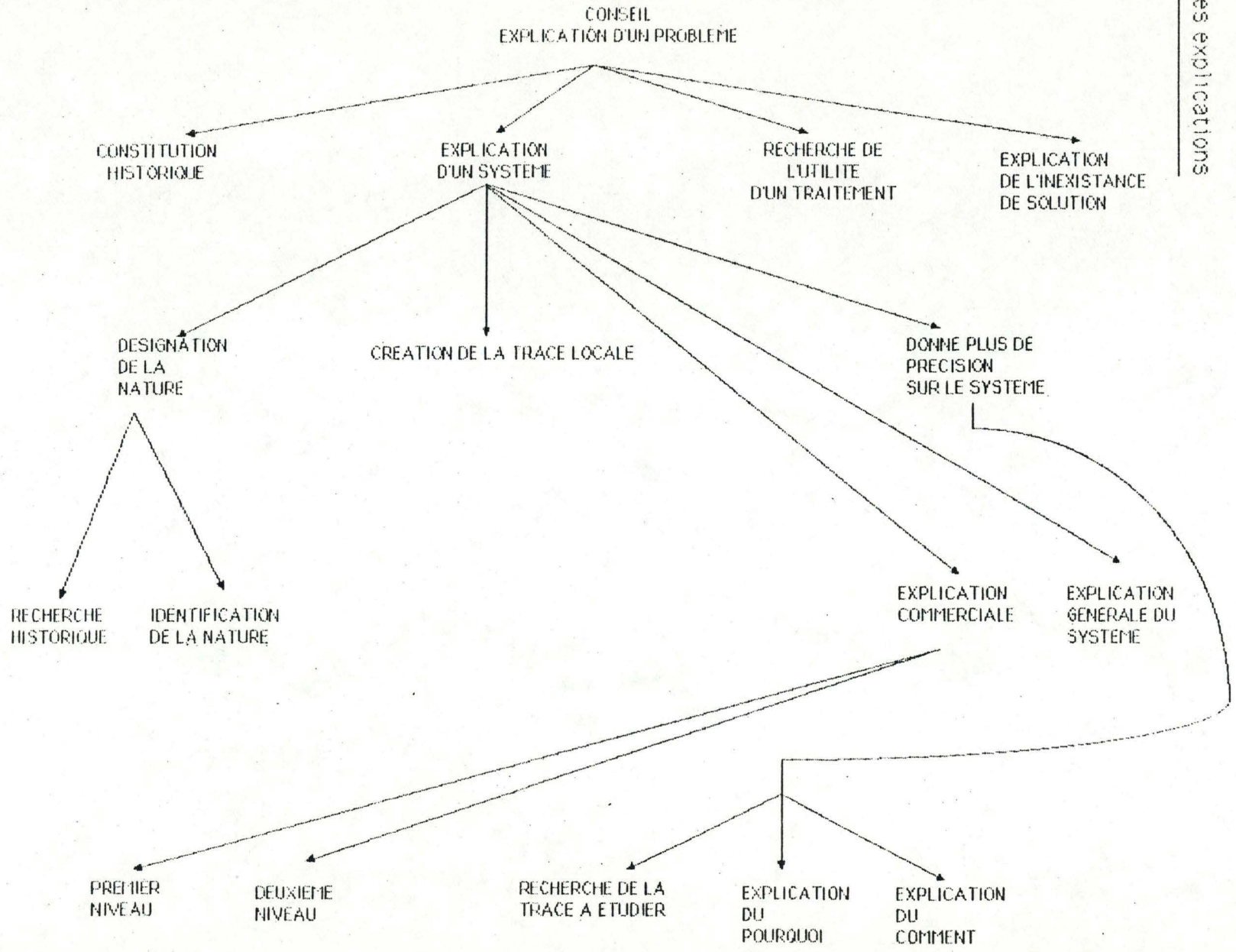
Avec la partie de la trace locale concernant un traitement et la structure statique, le système Conseil donne des explications sur les règles.

La question de l'utilisateur peut être :

- soit "Pourquoi cette règle concernant ce traitement ?" : après avoir identifié la règle appliquée au traitement dans la trace, le système sort à l'écran la phrase concernant l'état de la règle. Si cette règle a des enfants, ils sont également présentés à l'écran.

- soit "Comment cette règle concernant ce traitement ?" : après avoir trouvé le parent de cette règle et avoir identifié le parent de la règle appliquée au traitement dans la trace, le système sort à l'écran la phrase concernant l'état de la règle.

Ainsi un premier arbre "et" des tâches peut s'établir comme schématisé à la page suivante. Cet arbre est une étude détaillée du processus d'explication du système Conseil.


CONSEIL
EXPLICATION D'UN PROBLEME
CONSTITUTION HISTORIQUE
EXPLICATION D'UN SYSTEME
RECHERCHE DE L'UTILITE D'UN TRAITEMENT
EXPLICATION DE L'INEXISTANCE DE SOLUTION
DESIGNATION DE LA NATURE
CREATION DE LA TRACE LOCALE
DONNE PLUS DE PRECISION SUR LE SYSTEME
RECHERCHE HISTORIQUE
IDENTIFICATION DE LA NATURE
EXPLICATION COMMERCIALE
EXPLICATION GENERALE DU SYSTEME
PREMIER NIVEAU
DEUXIEME NIVEAU
RECHERCHE DE LA TRACE A ETUDIER
EXPLICATION DU POURQUOI
EXPLICATION DU COMMENT

VI.2. IMPLEMENTATION

Dans cette section, nous exposons en quelques mots, le langage PROLOG et la manière de spécifier les procédures. Dans un deuxième temps, nous présentons les types d'objets et les relations existantes entre chaque type d'objet. Ensuite nous décrivons la base de données ainsi que les spécifications des procédures du programme. En dernier lieu, nous finissons par les aspects non implémentés en raison du manque de temps.

VI.2.1. Introduction

L'implémentation de notre prototype est écrite en PROLOG. Ce langage semble adéquat à notre application car il nous permet de résoudre des problèmes qui peuvent s'exprimer en termes d'objets et de leurs relations.

C'est un langage qui a été conçu par le groupe de recherche en Intelligence Artificielle de Marseille, avec à sa tête Alain Colmerauer, dans le cadre de recherche sur la compréhension du langage naturel.[CLO84]

Un système PROLOG utilise un démontreur de théorèmes basé sur le Principe de Résolution de Robinson [ROB68] pour clauses de Horn.

En raison de la nature de PROLOG, la manière de spécifier les procédures n'est pas classique. En effet, dans d'autres langages, il y a les paramètres de type argument et les paramètres de type résultat. En PROLOG, langage logique, un paramètre peut être de deux types : argument ou résultat. Ce n'est qu'à l'appel de la procédure que l'on peut distinguer le type du paramètre. Si celui-ci est instancié alors le paramètre est de type argument, sinon il est de type résultat.

Nous spécifions nos procédures de la manière suivante.[DEV87]

procédure p (P1,P2,...,Pn).

a) déclaration

P1 est du type T1,
P2 est du type T2,
Pn est du type Tn.

où type Ti est soit de type entier, de type caractère, de type booléen ou de type liste de termes.

Si les paramètres sont soumis à des préconditions, elles sont spécifiées à cet endroit.

b) La relation

La procédure p détermine si la relation R qui y est associée est établie entre les paramètres P1,P2,...,Pn.

c) Les conditions d'application

C'est la description des emplois possibles de la procédure.

Pour chaque paramètre, nous retiendrons 3 formes possibles :

- instancié (ground) : c'est-à-dire que le paramètre est instancié à un terme ne contenant pas de variable.
- variable (var) : c'est-à-dire que le paramètre est libre, n'est pas instancié.
- instancié ou variable (any) : c'est-à-dire que le paramètre est soit instancié, soit libre.

Par 'paramètre instancié', nous entendons paramètre auquel nous avons assigné une valeur.
Par 'paramètre libre', nous entendons paramètre auquel nous n'avons assigné aucune valeur.
Si dans une procédure nous avons A = 8 et
- si A est assigné à 5 alors la procédure renvoit la valeur "FALSE".
- si A est libre alors A à la valeur 8.

d) Les effets de bord

Cette partie sera utilisée pour les aspects non logiques tel que les entrées/sorties.

VI.2.2. Stratégie globale

Dans Conseil, avec l'ajout des critères commerciaux, l'approche adoptée est la recherche de type en profondeur d'abord quand il s'agit de trouver tous les systèmes techniquement corrects, et de type heuristique quand il s'agit de trouver le ou les systèmes commerciaux.

Il y a 4 types de connaissances dans Conseil :
- la connaissance relative aux traitements
- la connaissance relative aux mécanismes de raisonnement pour sélectionner un ensemble de traitements constituant un système de peinture techniquement correct
- la connaissance relative à la fonction d'évaluation d'un système
- la connaisance relative aux mécanismes d'explication.

Les deux premiers types de connaissances sont représentés par un ensemble de règles PROLOG qui diffèrent par le fait que celles représentant la connaissance relative aux traitements ont pour paramètres des constantes alors que dans la deuxième catégorie, apparaissent surtout des variables. Le premier type de connaissance sert comme base de connaissances, ce qui explique pourquoi la pluspart de ces paramètres sont instanciés. Quant au deuxième type, il essaie

de sélectionner un ensemble de traitements en utilisant la base de connaissances. Par conséquent, ces paramètres sont des variables.

Le troisième type de connaissance ne nous sert pas lorsque l'on veut tous les systèmes techniquement corrects pour le problème posé. Il est utile dès que l'on veut avoir les meilleurs systèmes du point de vue économique. Nous appliquons la connaissance de type 3 quand Conseil a le choix de sélectionner plus d'un traitement. Cette connaissance nous permet de choisir le ou les meilleurs traitements et par conséquent d'établir un échelle de qualité entre les systèmes générés.

Le quatrième type de connaissance est propre au domaine d'explication. Lors de la recherche de systèmes de peinture, cette connaissance n'est pas utilisée. Elle nous donne comme information, ce qu'il faut expliquer (c'est-à-dire la structure statique) et la manière de l'expliquer (c'est-à-dire un ensemble de phrases en francais).

VI.2.3. Définition des types d'objets

VI.2.3.1. Les types d'objets

Les types d'objets manipulés dans le programme PROLOG proviennent directement de la modélisation par espace d'états; ils ont été réorganisés de manière légèrement différente pour faciliter l'implémentation PROLOG. La structure des objets n'a pas changé avec l'implémentation de la recherche commerciale et du système d'explication. Nous y avons ajouté d'autres informations utiles pour la bonne marche du programme. La définition et les caractéristiques des objets sont décrites ci-dessous :

VI.2.3.1.1. Les objets associés aux traitements

TRAITEMENT : une préparation du support ou une application d'un produit de la gamme Trimetal.

COMPOSITION : description de la composition chimique en termes de 4 composants.

COMPOSANT : solvants ou liants ou pigments entrant dans la composition d'un traitement, ou encore taux de concentration de pigments en volume (PVC).

NOM DU TRAITEMENT : identifiant d'un traitement, c'est-à-dire son nom ou son appellation commerciale.

EFFET DE BORD : renseignements économiques relatifs à un traitement. Il est composé de 12 DESCRIPTIFs.

DESCRIPTIF : prix au mètre carré ou temps de séchage ou odeur ou composition ou base ou promotion ou facilité d'application ou ancienneté ou état du stock ou qualité ou mode d'application au rouleau ou au pistolet.

PRECONDITION : VECTEUR D'ETAT représentant un état du support avant application du traitement.

POSTCONDITION : VECTEUR D'ETAT représentant un état du support après application du traitement.

VI.2.3.1.2. Les objets associés à la recherche des systèmes

VECTEUR GENERAL : couple composé d'un VECTEUR D'ETAT et d'un VECTEUR D'ETAT COMMERCIAL.

VECTEUR D'ETAT : description d'un support à peindre, dans un état donné en terme de 9 COMPOSANTES.

COMPOSANTES : nature ou forme ou environnement ou caractéristiques spéciales ou aspect ou état de surface ou résistance ou protections ou dernier traitement vers le haut d'un support.

VECTEUR D'ETAT COMMERCIAL : description d'un support à peindre en terme de 12 COMPOSANTEs ECONOMIQUEs dont une a comme valeur l'évaluation de l'état donné par rapport aux autres.

COMPOSANTE ECONOMIQUE : prix au mètre carré ou temps de séchage ou odeur ou composition ou base ou promotion ou facilité d'application ou ancienneté ou qualite ou mode d'application au rouleau ou au pistolet ou évaluation de l'état.

CLASSE : composé d'une valeur de composante et de 12 DESCRIPTIFs.

COEFFICIENT D'IMPORTANCE : description des coefficients d'importance des 12 DESCRIPTIFs.

COEFFICIENT DE PONDERATION : description des coefficients de pondération des 12 DESCRIPTIFs.

QUESTION : description en fonction d'un descriptif correspondant à un critère extra-technique avec question en termes de 4 CARACTERISTIQUEs.

CARACTERISTIQUE : quest en français ou valeur ou caract ou évaluation.

VI.2.3.1.3. Les objets associés au processus d'explication

REGLE : nom de règle du programme PROLOG.

PHRASE : ensemble de phrases en français.

ENFANT : composé d'une règle parent et d'une liste de règles qui ont comme caractéristique d'être enfant de la règle parent.

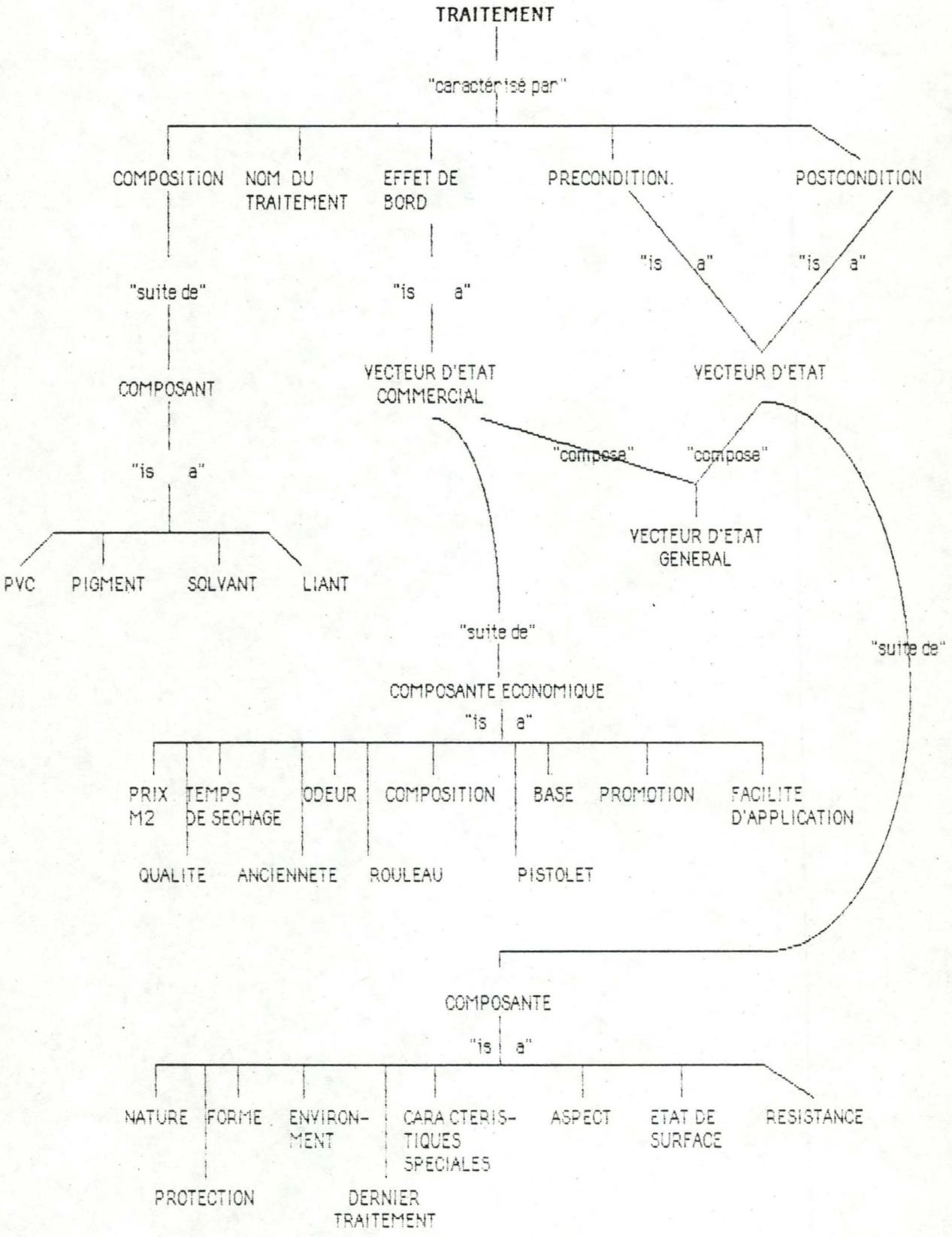
PARENT : composé de 2 règles, la première étant l'enfant de la deuxième.

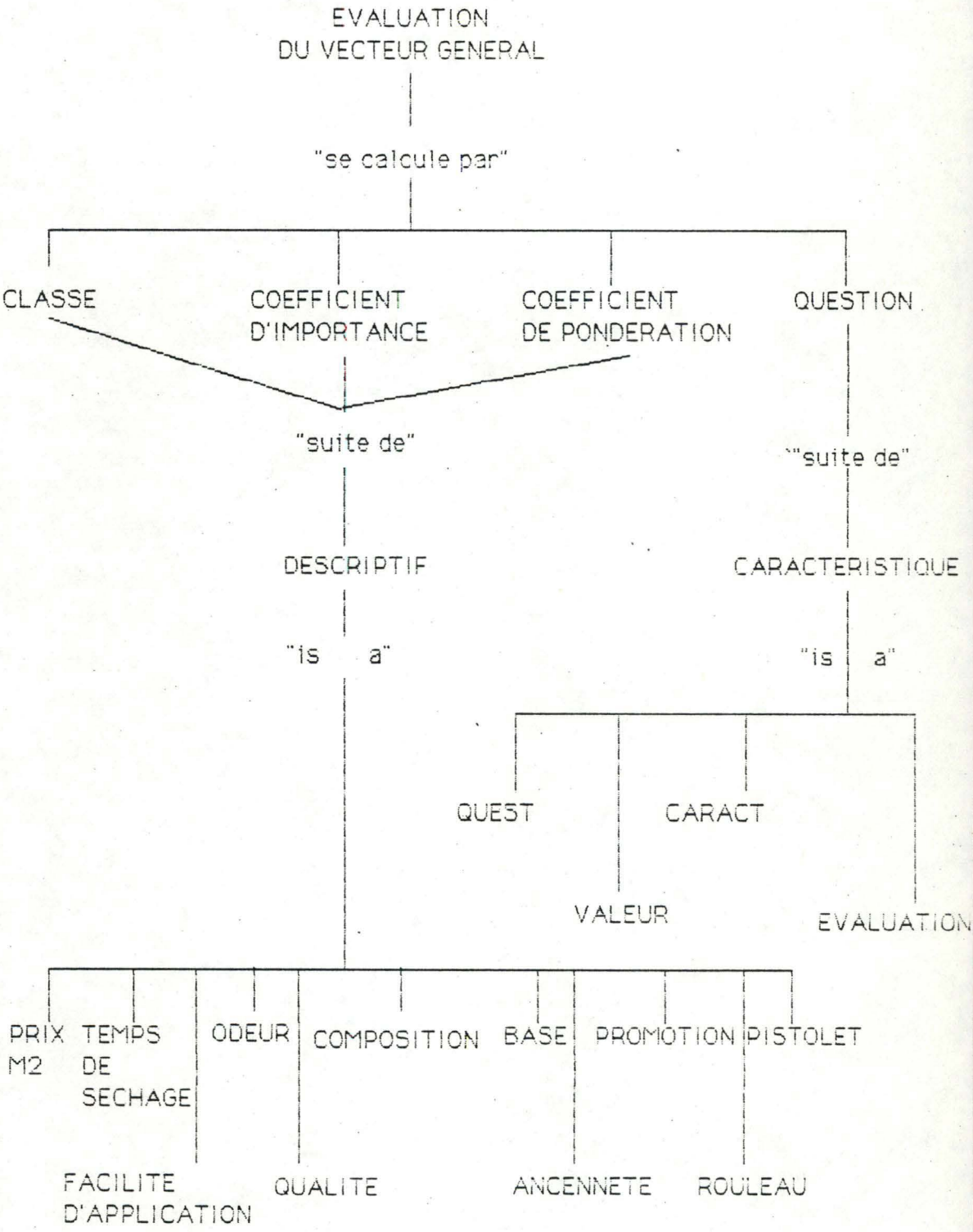
FRANCAIS : composé d'une identifiant à une règle et d'une phrase.

IDENTIFIANT D'UNE REGLE : composé
- d'un nom de règle
- du numéro de la règle ayant réussi ou de la mention de 'échec'
- de données locales et globales et des paramètres concernant la règle (cfr V.5.3.1.).

IDENTIFICATION D'UN SYSTEME : composé
- d'un état
- du numéro de la catégorie
- de données locales et globales et des paramètres concernant le système.

Le traitement et ses objets




EVALUATION
DU VECTEUR GENERAL
"se calcule par"
CLASSE
COEFFICIENT
D'IMPORTANCE
COEFFICIENT
DE PONDERATION
QUESTION
"suite de"
"suite de"
DESCRIPTIF
CARACTERISTIQUE
"is a"
"is a"
QUEST
CARACT
VALEUR
EVALUATION
PRIX
M2
TEMPS
DE
SECHAGE
ODEUR
COMPOSITION
BASE
PROMOTION
PISTOLET
FACILITE
D'APPLICATION
QUALITE
ANCENNETE
ROULEAU

VI.2.3.2. Primitives de manipulation associées à ces types d'objets

Les spécifications de ces primitives sont énoncées aux sections VI.2.4 et VI.2.5.

Types d'objets	Procédure PROLOG
traitement	POST PRE COMP EXTPVC EXTPIG EXTLIANT EXTSOL EFFETDEBORD
Solvant	ECHELAGR
Liant	ECHELTHERM SAPONIFIABLE
Pigment	COMPZINC
Vecteur d'état	EXTNAT EXTENV EXTFORME EXTASP...(cfr [DEC86])
Vecteur d'état commercial	EXTEV EXTPROM EXTFAC EXTNPRO EXTODEUR EXTANC EXTPIS EXTROU EXTTSECH EXTCOMP EXTQUAL EXTSOL

VI.2.3.3. Les types d'objets génériques

VI.2.3.3.1. Structure

Les types d'objets génériques sont des abstractions dont le rôle est d'accroître la lisibilité et la modifiabilité de l'implémentation de CONSEIL; ils sont définis de la façon suivante :

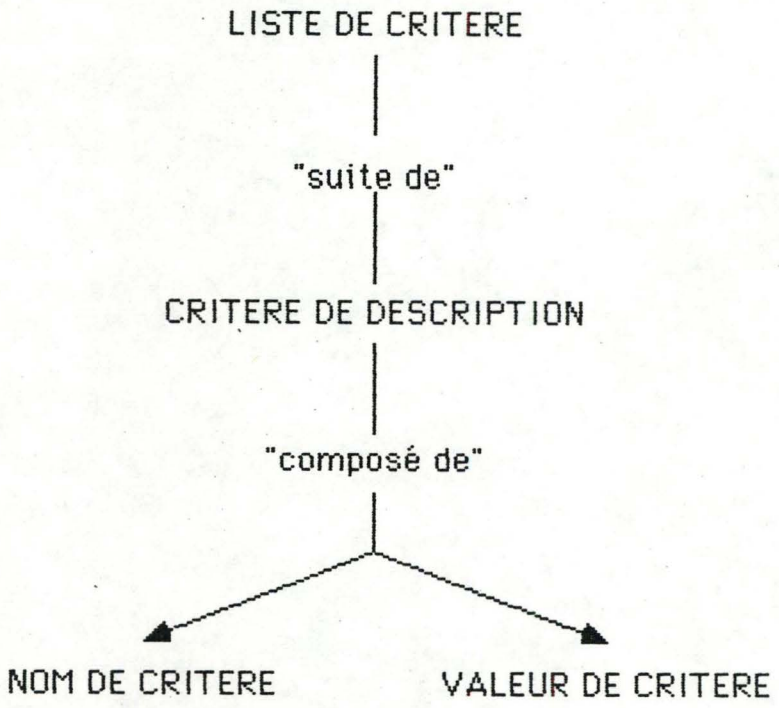
LISTE DE CRITERE : suite de CRITERES DE DESCRIPTION.

CRITERE DE DESCRIPTION : couple composé d'un nom de composante et de sa valeur.

NOM DE CRITERE : premier des deux éléments du couple CRITERE DE DESCRIPTION.

VALEUR DE CRITERE : second des deux éléments du couple CRITERE DE DESCRIPTION.

Les types génériques sont structurés selon le schéma suivant :



VI.2.3.3.2. Structures de représentations internes

a) Définition de nouvelles structures PROLOG.

A côté des structures existantes de liste et d'atome, nous avons, pour des raisons de modifiabilité, défini de nouvelles structures utilisées pour l'implémentation des types d'objets et des types génériques.

CRITERE_DE_DESCRIPTION : liste de deux termes.

LISTE_DE_CRITERE : liste de CRITERE_DE_DESCRIPTIONs.

LISTE_DE_VALEUR : liste de VALEURs.

LISTE_DE_REGLE : liste de NOM_DE_REGLEs.

PHRASES : liste de PHRASE.

PHRASE : liste de MOT.

NOM_DE_CRITERE : atome.

TRAITEMENT : atome.

NOM DE REGLE : atome.

VALEUR : atome.

MOT : atome.

LISTE_VIDE : représentation d'une liste ne contenant aucun terme.

b) Structures de représentation des types d'objets

Pour chaque occurrence d'un TRAITEMENT, la relation "caractérisé par" est représentée par quatre faits PROLOG. Dans chacun de ceux-ci, le premier paramètre est un NOM DE TRAITEMENT qui représente NOM DE TRAITEMENT et identifie l'occurrence de TRAITEMENT.

Voici ces quatre faits sur lesquels nous reviendrons plus longuement à la section VI.2.4.1.

```
COMP (NT, COMPOSITION),
PRE (NT, PRECONDITION),
POST (NT, POSTCONDITION),
EFFETDEBORD (NT, DESCRIPTIF).
```

COMPOSITION, VECTEUR D'ETAT et VECTEUR D'ETAT COMMERCIAL sont représentés par des LISTEs_DE_CRITERE et, par conséquent, COMPOSANT et COMPOSANTE sont implémentés sous forme de CRITEREs_DE_DESCRIPTION.

VECTEUR D'ETAT GENERAL est représenté par un couple de LISTE_DE_CRITEREs.

c) Structures de représentation des types génériques

Les LISTEs DE CRITERE sont implémentées sous forme de LISTEs_DE_CRITERE et les CRITEREs DE DESCRIPTION sous forme de

CRITERE_DE_ DESCRIPTION.

Un NOM DE CRITERE est représenté par un NOM_DE_ CRITERE.

Une VALEUR DE CRITERE est représentée soit par une VALEUR si elle contient une seule valeur, soit par une LISTE_DE_VALEUR si elle en contient plusieurs.

VI.2.4. Architecture de la base de faits

Nous présentons dans cette section la base de faits concernant la recherche des systèmes de peinture, et celles concernant l'explication du processus de raisonnement du système Conseil.

VI.2.4.1. La recherche des systèmes de peinture

Nous faisons ici référence aux informations statiques concernant les traitements et la fonction d'évaluation de ces traitements. Elles sont indispensables lors de la phase de sélection d'un traitement candidat, de vérification de compatibilité avec la suite déjà construite, de calcul du nouvel état et d'évaluation du nouveau système de peinture; elles portent sur les COMPOSANTs, COMPOSANTEs ECONOMIQUEs, DESCRIPTIFs, PRECONDITIONs et POSTCONDITIONs d'un TRAITEMENT.

Les faits représentant ces connaissances sont organisés en quatre parties : la première est issue directement de la représentation interne du type d'objet TRAITEMENT. La deuxième et la troisième classe de faits sont exposée dans [DEC86]. La quatrième classe est utilisée pour l'évaluation des systèmes de peinture.

1. Première classe de faits

Pour les règles POST, PRE, COMP, le lecteur se reportera à [DEC86].

- effetdebord(Traitement,Vecteur).

Traitement est de type NOM_DE_TRAITEMENT,
Vecteur est de type LISTE_DE_CRITERES.

relation : la procédure détermine si Vecteur est la représentation interne de l'effet de bord du traitement dont la représentation interne est TRAITEMENT.

condition d'application :
in(any,any) : out(ground,ground).

2.3. Deuxième et troisième classe de faits : le lecteur se reportera à [DEC86].

4. Quatrième classe de faits

- coeffimp(L).

  L est de type LISTE_DE_CRITERES.

relation : la procédure détermine si chacun des noms de critères est la représentation interne d'un descriptif, et si la valeur du critère correspondant est la représentation interne de l'importance de ce descriptif par rapport aux autres.

condition d'application :
in(any) : out(ground).

- coeffpond(L).

  L est de type LISTE_DE_CRITERE.

relation : la procédure détermine si chacun des noms de critères est la représentation interne d'un descriptif, et si la valeur du critère correspondant est la représentation interne de la pondération de ce descriptif par rapport aux autres.

condition d'application :
in(any) : out(ground).

- question(L).

  L est de type LISTE_DE_CRITERE.

relation : la procédure détermine si L est la représentation interne d'une caractéristique avec question.

condition d'application :
in(any) : out(ground).

- classe(A,L).

  A est de type VALEUR.
  L est de type LISTE_DE_CRITERE.

relation : la procédure détermine si chacun des noms de critères est la représentation interne d'un descriptif, et si la valeur du critère correspondant est la représentation interne de la plus petite valeur d'un nom de critère identique à un traitement ayant comme caractéristique A.

VI.2.4.2. L'explication des systèmes de peinture

Ces faits sont indispensables lors d'une explication d'un système de peinture pour un utilisateur.

Les faits représentant ces connaissances sont organisés en deux classes : la première est la connaissance dite statique du mécanisme de raisonnement du système expert CONSEIL, et la deuxième traite des explications données à l'utilisateur.

1. Première classe de faits

- enfant(R,LR).

  R est de type NOM_DE_REGLE,
  LR est de type LISTE_DE_REGLE.

relation : la procédure détermine si LR est la représentation interne de ENFANT de R.

condition d'application :
in(any,any) : out(ground,ground).

- parent(RE,RP).

  RE et RP sont de type NOM_DE_REGLE.

relation : la procédure détermine si RE est la représentation interne de PARENT de RP.

condition d'application :
in(any,any) : out(ground,ground).

2. Deuxième classe de faits

- français([R,N,Do],PHR).

  R est de type NOM_DE_REGLE,
  N est de type VALEUR,
  Do est de type LISTE_DE_VALEUR,
  PHR est de type PHRASES.

relation : la procédure détermine si [R,N,Do] est la représentation interne d'un identifiant d'une règle, et si PHR est la représentation interne d'une phrase, celle-ci correspondant à l'identifiant de la règle.

condition d'application :
in([any,any,any],any) : out([ground,ground,ground],ground).

- expl1(N,Cat,S1,S2,S3,PHr).

  N est de type ATOME,

CAT est de type ATOME,
S1,S2,S3 sont de type LISTE_DE_VALEUR,
PHr est de type PHRASES.

relation : la procédure détermine si [N,CAT,S1,S2,S3] est la représentation interne d'une identification d'un système et si PHr est la représentation interne d'une phrase, celle-ci correspondant à l'identification de la règle.

condition d'application :
in(any,any,any,any,any,any) : out(ground,ground,ground,ground,ground, ground).

- traduction(N,R,Phr)

N est un entier,
R est de type NOM_DE_REGLE,
Phr est de type PHRASE.

relation : la procédure détermine si R est la représentation interne de N et si Phr est la présentation de R en français.

condition d'application :
in(any,any,any) : out(ground,ground,ground).

VI.2.5. Architecture de la base de règles

La base de règles se divise en 3 parties :

- les règles de haut niveau utiles pour la recherche de systèmes commerciaux et pour donner des explications à l'utilisateur.

- les primitives de bas niveau où l'on retrouve les procédures de manipulation de listes, de listes de critères, de fichiers et ainsi que les primitives d'entrée/sortie.

- les primitives se rapportant à l'interface clavier/écran.

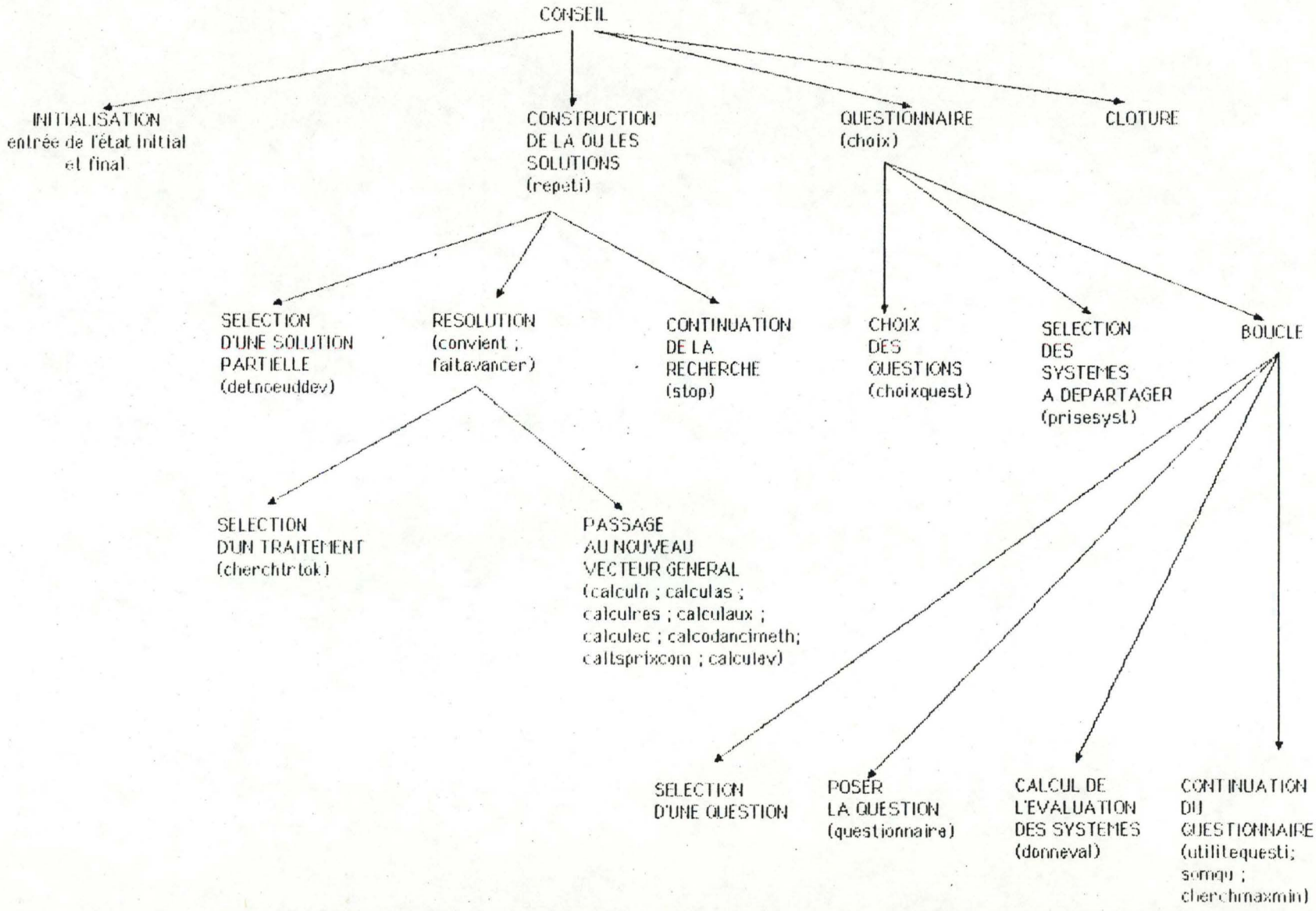
VI.2.5.1. Les procédures de haut niveau

Les procédures de haut niveau constituent le coeur du programme. En effet, elles traduisent le mode de raisonnement suivi.

VI.2.5.1.1. Procédure pour la recherche de systèmes économiques

Le schéma présenté ci-après représente le raisonnement suivi dans le processus de conseil en peinture. Ce mode de raisonnement est décomposé en sous-tâches selon un ordre chronologique.

Chaque arbre se trouvant à gauche d'un autre doit être exécuté avant, sauf s'il s'agit d'un arbre OU.


CONSEIL
INITIALISATION
entrée de l'état initial
et final
CONSTRUCTION
DE LA OU LES
SOLUTIONS
(repeti)
QUESTIONNAIRE
(choix)
CLOTURE
SELECTION
D'UNE SOLUTION
PARTIELLE
(detnoeuddev)
RESOLUTION
(convient ;
faitavancer)
CONTINUATION
DE LA
RECHERCHE
(stop)
CHOIX
DES
QUESTIONS
(choixquest)
SELECTION
DES
SYSTEMES
A DEPARTAGER
(prisesyst)
BOUCLE
SELECTION
D'UN TRAITEMENT
(cherchtrtok)
PASSAGE
AU NOUVEAU
VECTEUR GENERAL
(calculn ; calculas ;
calculres ; calculaux ;
calculec ; calcodancimeth;
calltsprixcom ; calculev)
SELECTION
D'UNE QUESTION
POSER
LA QUESTION
(questionnaire)
CALCUL DE
L'EVALUATION
DES SYSTEMES
(donneval)
CONTINUATION
DU
QUESTIONNAIRE
(utilitequesti;
somqu ;
cherchmaxmin)

1. Construction de la ou les solutions

- procédure repeti(X).

  X est un entier.

condition d'application :
in(ground) : out(ground).

effet de bord : la procédure a pour effet de trouver au maximum un nombre X de solutions économiques (il est possible qu'il n'existe pas autant de solutions) et d'écrire dans la base de régles les règles
 -'liresolterm' représentant les solutions terminales,
 -'liresol' représentant des solutions non terminales.

1.1. Sélection d'une solution partielle

L'objectif est, ayant les solutions non terminales avec leurs évaluations, de sélectionner la solution non terminale la plus intéressante pour l'utilisateur.

- procédure detnoeuddev(L,N).

  L est une liste dont les éléments sont de type CRITERE_DE_ DESCRIPTION
  N est un couple de type LISTE_DE_CRITEREs.

pré : L est une liste de couples triés selon la valeur de critère de nom critère 'evalu'. Ce critère de description se trouve dans une liste de critères du deuxième élément du couple.

relation : la procédure détermine si N est le premier élément de la liste L.

condition d'application :
in(ground,any) : out(ground,ground).
in(any,ground) : out(ground,ground).

1.2. Continuation de la recherche

L'objectif est de savoir si cela vaut la peine de continuer à rechercher des solutions terminales.

- procédure stop(X,L1,L2).

  X est un entier,
  L1, L2 sont des listes dont les éléments sont de type CRITERE_DE_ DESCRIPTION.

pré : L1 et L2 sont des listes de couples avec une valeur de critère

de nom de critère 'evalu' représentant leur évaluation.

relation : la procédure détermine - si L2 est vide,
- ou si le nombre d'éléments de L1 est supérieur à X et si le minimun (par rapport à leurs valeurs d'évaluation) des éléments de L2 est supérieur au maximum des éléments de L1 (par rapport à leurs valeurs d'évaluation).

condition d'application :
in(ground,ground,any) : out(ground,ground,ground).

1.3 Résolution

1.3.1. Sélection d'un traitement

L'objectif est, ayant une solution partielle, de trouver un traitement utile et qui convienne.

- procédure cherchtrtok(Trt,Ed,Ec).

Ed,Ec sont de type CRITERE_DE_DESCRIPTION,
Trt est de type TRAITEMENT.

pré : Ed et Ec ont en commun les valeurs de critère des noms de critère 'nature','env' et 'forme'.

relation : la procédure détermine si le traitement de nom Trt fait avancer vers Ed et convient à Ec.

condition d'application :
in(any,ground,ground) : out(ground,ground,ground).

1.3.2. Passage au nouveau vecteur général

L'objectif est de passer d'un état courant à un autre avec l'apport d'un traitement. Cette partie calcule le nouveau vecteur d'état général.

- procédure calculn([N,N1], [Ec,Ec1], Trt, Aps, Ass, Rs, Pros, Ca, Esp, Asp, Rp, Prop, Dt).

N,N1,Ec,Ec1 sont de type LISTE_DE_CRITERE.
Trt, Aps, Ass, Rs, Pros, Ca, Esp, Asp, Rp, Prop, Dt sont des listes dont les éléments sont de type VALEUR.

relation : la procédure détermine si Ec est égal à un vecteur d'état dont: "aspect" a pour valeur le résultat de la procédure calculas;
"resist" " calculres;
"protect" " calculaux;
"applipart" " calculap;

"carspec" a pour valeur celle représentée par Ca;
"etatsurf" " Esp;
"derntrt" " Dt,
et si Ec1 est égal à un VECTEUR D'ETAT COMMERCIAL. Cette valeur est trouvée grâce à la procédure calculec.

condition d'application :
in(ground,var,ground,...,ground):out(ground,ground,ground,...,ground).

- procédure calculas(Asc,Ass,Asp,Asf).

Asc,Ass,Asp,Asf sont des listes de type VALEUR.

relation : la procédure détermine si Asc est égal à Asf + Asp + (Asf - Ass).

condition d'application :
in(any,ground,ground,ground) : out(ground,ground,ground,ground).

- procédure calculres(Resc,F,S,P,Perm).

F,S,P,Perm sont des listes dont les éléments sont de type VALEUR,
Resc est de type LISTE_DE_CRITERE.

relation : la procédure détermine si Resc est égal à F - P + (F - S).

condition d'application :
in(any,ground,ground,ground) : out(ground,ground,ground,ground).

- procédure calculaux(Perm,Supp,Moins,Pre,Res).

Perm,Supp,Moins,Pre,Res sont de type LISTE_DE_CRITERE.

relation : la procédure détermine si Res est égal à Pre + (Supp - Moins).

condition d'application :
in(ground,ground,ground,ground,any) : out(ground,ground,ground,ground,ground).

- procédure calculec(Ec1,Ec,Trt,N1).

Ec1,N1,Ec sont de type LISTE_DE_CRITERE,
Trt est de type TRAITEMENT.

relation : la procédure détermine si Ec1 est égal à la valeur d'un VECTEUR D'ETAT COMMERCIAL dont les valeurs de critères sont :
"sol" a pour valeur celle représentée par Trt + la valeur de critère de nom de critère "sol" de N1.
"Npr" a pour valeur celle représentée par 1 + la valeur de critère de nom de critère "Npro" de N1.

"prom" a pour valeur celle représentée par la somme des valeurs de critère de nom de critère "prom" de N1 et effet de bord de Trt.
"fac" a pour valeur celle représentée par "oui" ssi une des deux valeurs de critère de nom de critère "fac" de N1 et effet de bord de Trt est "oui".
"anci","odeur","maplir","maplip" ont pour valeur le résultat de la procédure calcodancimeth.
"tsech","prix","qual","comp" ont pour valeur le résultat de la procédure caltsprixcom.
"evalu" a pour valeur le résultat de la procédure calculev.

condition d'application :
in(any,ground,ground,ground) : out(ground,ground,ground,ground).

- procédure calcodancimeth (O,A,Pi,R,Eb,N1).

N1,Eb sont de type LISTE_DE_CRITERE,
O,A,Pi,R sont de type VALEUR DE CRITERE.

relation : la procédure détermine si O,A,Pi,R, c'est-à-dire les représentations internes des noms de critère odeur, anci, maplip, maplir, sont égaux aux valeurs suivantes.
O = "o" si une des deux valeurs de nom de critère "odeur" de N1 et Eb est "o".
A = "n" si une des deux valeurs de nom de critère "anci" de N1 et Eb est "n".
Pi = "n" si une des deux valeurs de nom de critère "maplip" de N1 et Eb est liste vide.
R = "n" si une des deux valeurs de nom de critère "maplir" de N1 et Eb est liste vide.

condition d'application :
in(any,any,any,any,ground,ground) : out(ground,ground,ground,ground, ground).

- procédure caltsprixcom(Ec,Ts,P,Co,Q,Npr,Eb,N1).

Ec,Eb,N1 sont de type LISTE_DE_CRITERE,
Npr,Ts,P,Co,Q sont de type VALEUR DE CRITERE.

relation : la procédure détermine si Ts, P,Co, Q ont de bonnes valeurs selon les différentes valeurs de Npr,Eb,N1.

condition d'application :
in(any,any,any,any,any,ground,ground,ground) : out(ground, ground, ground, ground, ground, ground, ground, ground,

- procédure calculev(Ec,Ev,L).

Ec est de type LISTE_DE_CRITERE,
L est une liste dont les éléments sont de type VALEUR,
Ev est de type VALEUR DE CRITERE.

relation : la procédure détermine si Ev est égal à la valeur d'évaluation du VECTEUR D'ETAT Ec sur base de L. Si Ec est l'état terminal, alors Ev est égal à l'évaluation du système de peinture. Par contre si Ec n'est pas un état final, alors Ev est égal à l'évaluation du système de peinture partiel plus l'évaluation d'un traitement futur.

condition d'application :
in(ground,any,ground) : out(ground,ground,ground).

2. le questionnaire

L'objectif est, après avoir fait la recherche, de distinguer, en posant des questions à l'utilisateur, les systèmes terminaux.

- procédure choix(L,X,R).

L,R sont des listes dont les éléments sont de type LISTE_DE_CRITERE, X est un entier.

pré : chaque élément de L est la représentation interne d'un système de peinture répondant au même problème.

relation : la procédure détermine - si R est égal à L lorsque le nombre d'éléments de L est inférieur à X,
- ou si R est égal à un sous-ensemble de X éléments de L différenciés grâce à des questions posées à l'utilisateur.

condition d'application :
in(any,ground,ground) : out(ground,ground,ground).
in(ground,any,any) : out(ground,ground,ground).

2.1. Le choix des questions

L'objectif est de ne poser à l'utilisateur que les questions susceptibles de distinguer les systèmes de peinture entre eux.

- procédure choixquest(Ltri,Quest).

Ltri,Quest sont des listes dont les éléments sont de type LISTE_DE_CRITERE.

relation : la procédure détermine si Quest est égal à des listes de critères utiles pour la différenciation des éléments de Ltri entre eux. Cette différenciation se fait grâce aux valeurs de critères des éléments de Ltri. Quest est trié sur la valeur de critère de nom de critère "evalu" de ces éléments d'une façon décroissante.

condition d'application :
in(ground,any) : out(ground,ground).

2.2. Sélection des systèmes à départager

L'objectif est de sélectionner les systèmes de peinture que la fonction d'évaluation n'a pas su départager. Ces systèmes pourront être départagés grâce aux questions.

- procédure prisesyst(L,Ens,Emax,Emin).

L,Ens sont des listes dont les éléments sont de type LISTE_DE_CRITERE,
Emax,Emin sont des entiers.

relation : la procédure détermine si Ens contient l'ensemble des éléments de L dont la valeur de critère de nom de critère "evalu" est comprise dans l'intervalle [Emax,Emin].

condition d'application :
in(ground,ground,ground,any) : out(ground,ground,ground,ground).
in(any,ground,ground,ground) : out(ground,ground,ground,ground).

2.3. Boucle

2.3.1. Poser la question à l'utilisateur

Le but est que l'utilisateur donne son avis afin de différencier les systèmes de peinture. Cette partie, en lui posant une question, lui permet de donner son avis.

- procédure questionnaire(Qu,Val,Reponse).

Qu est de type LISTE_DE_CRITERE,
Val est un entier,
Reponse est un caractère.

pré : Qu est la représentation interne d'une caractéristique.

relation : La procédure détermine si Val est égal à la valeur de critère de nom de critère "evalu" de Qu et si Reponse est égal à la réponse donnée par l'utilisateur.

condition d'application :
in(ground,any,any) : out(ground,ground,ground).

effet de bord : à l'écran, apparaît une question et l'utilisateur doit rentrer une réponse au clavier.

2.3.2. Calcul de l'évaluation des systèmes.

L'objectif est, au moyen de la réponse donnée par l'utilisateur, de mettre à jour l'évaluation des systèmes de peinture.

- procédure donneval(L,Reponse,Qu,Val,Lval).

L,Lval sont des listes dont les éléments sont de type LISTE_DE_CRITERE,
Qu est de type LISTE_DE_CRITERE,
Reponse est un caractère,
Val est un atome.

pré : les éléments de Qu sont la représentation interne d'une caractéristique.

relation : la procédure détermine si Lval contient tous les éléments de L dont la valeur de critère de nom de critère "evalu" a été augmentée de la valeur de critère de nom de critère "evalu" de Qu :
- si Reponse = "o"
- et si la valeur de critère de nom de critère de la valeur de critère de nom de critère "car" de Qu de L est égal à Val.

condition d'application :
in(ground,ground,ground,ground,any) : out(ground, ground, ground, ground, ground).

2.3.3. Continuation du questionnaire

L'objectif est de vérifier si cela vaut la peine de continuer à poser des questions pour différencier les systèmes de peinture.

- procédure utilitéquesti(Min,Max,Quest,R).

Min,Max sont des entiers,
Quest est une liste dont les éléments sont de type LISTE_DE_CRITERE,
R est un booléen.

Pre : R a la valeur "stop" ou "continuer".

relation : la procédure détermine si R est égal - à "stop" quand la somme de (Max + la somme des valeurs de critère de nom de critère "evalu"de Quest) est inférieure ou égale à Min.
- à "continuer"dans les autres cas.

condition d'application :
in(ground,ground,ground,any) : out(ground,ground,ground,ground).

- procédure somqu(Quest,Res).

  Quest est une liste dont les éléments sont de type LISTE_DE_CRITERE,
  Res est un entier.

relation : la procédure détermine si Res est égal à la somme des valeurs de critère de nom de critère "evalu" des éléments de la liste Quest.

condition d'application :
in(ground,any) : out(ground,ground).

- procédure cherchmaxmin(X,Ltri,Max,Min).

  X,Max,Min sont des entiers,
  Ltri est une liste dont les éléments sont de type LISTE_DE_CRITERE.

Pré : Ltri est trié de manière croissante sur la valeur de critère de nom de critère "evalu".

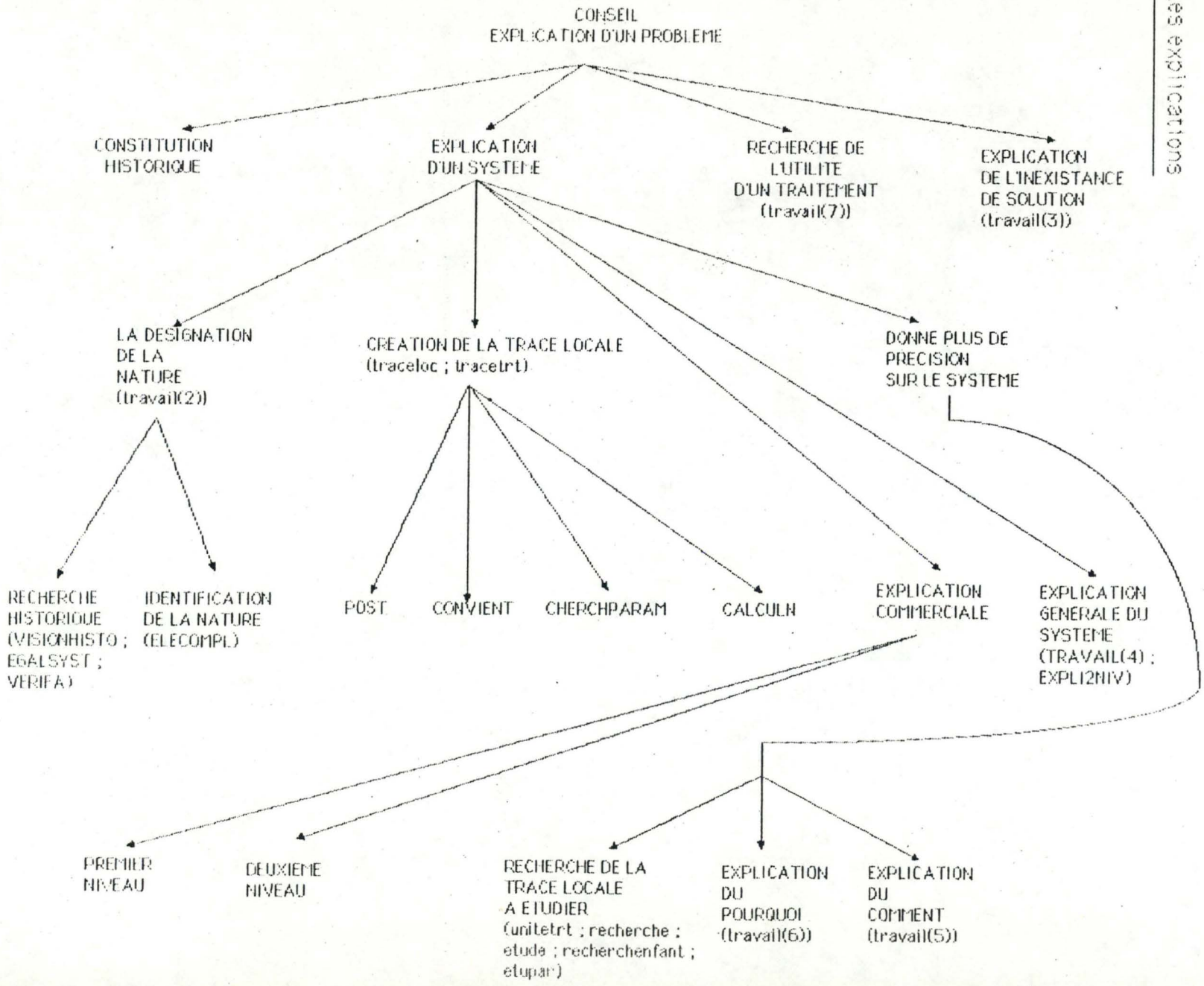
relation : la procédure détermine - si Max est égal à la valeur de critère de nom de critère "evalu" du Xième élément de la liste Ltri,
- et si Min est égal à la valeur de critère de nom de critère "evalu du (Xième + 1) élément de Ltri.

condition d'application :
in(ground,ground,any,any) : out(ground,ground,ground,ground).

VI.2.5.1.2. Procédure pour l'explication du système expert

Le schéma présenté ci-après représente le raisonnement suivi dans le processus d'explication de conseil en peinture. Ce mode de raisonnement est décomposé en sous-tâches selon un ordre chronologique.

A propos de la construction historique, cette tâche se fera lors de l'éxécution de la recherche du ou des systèmes de peinture. En plaçant à certains endroits clés du programme des instructions d'écriture (cfr V.5.3.2.) dans un fichier, les informations nécessaires à la construction de l'historique seront enregistées.


CONSEIL
EXPLICATION D'UN PROBLEME
CONSTITUTION HISTORIQUE
EXPLICATION D'UN SYSTEME
RECHERCHE DE L'UTILITE D'UN TRAITEMENT (travail(7))
EXPLICATION DE L'INEXISTANCE DE SOLUTION (travail(3))
LA DESIGNATION DE LA NATURE (travail(2))
CREATION DE LA TRACE LOCALE (traceloc ; tracetrt)
DONNE PLUS DE PRECISION SUR LE SYSTEME
RECHERCHE HISTORIQUE (VISIONHISTO ; EGALSYST ; VERIFA)
IDENTIFICATION DE LA NATURE (ELECOMPL)
POST
CONVIENT
CHERCHPARAM
CALCULN
EXPLICATION COMMERCIALE
EXPLICATION GENERALE DU SYSTEME (TRAVAIL(4) ; EXPLI2NIV)
PREMIER NIVEAU
DEUXIEME NIVEAU
RECHERCHE DE LA TRACE LOCALE A ETUDIER (unitetrt ; recherche ; etude ; recherchenfant ; etupar)
EXPLICATION DU POURQUOI (travail(6))
EXPLICATION DU COMMENT (travail(5))

1. Désignation de la nature d'un système

L'objectif de cette partie est de renseigner l'utilisateur sur la nature du système de peinture qu'il a proposé. Cet objectif utilise l'information contenue dans l'historique.

Ces informations sont contenues dans des fichiers. Chaque catégorie a un fichier. Exemple : la catégorie numéro 1 est enregistré dans le fichier CAT1.PRO.

Les systèmes terminaux ou les systèmes partiels de peinture sont mis dans leurs catégories respectives les uns à la suite des autres.

Au moyen de cette information, le système expert sort une phrase à l'utilisateur l'aidant à mieux situer son système de peinture dans la recherche faite précédemment.

- procédure travail(X).

X est un entier.

pré : X vaut 2.

effet de bord : la procédure a pour effet de sortir une phrase expliquant la nature du système de peinture proposé par l'utilisateur.

1.1. La recherche dans l'historique

L'objectif est de trouver dans une catégorie le système se rapprochant le plus possible du système donné par l'utilisateur.

- procédure visionhisto(S, Cat, Syshisto, Systrendu).

S, Syshisto, Systrendu sont des listes dont les éléments sont de type TRAITEMENT.
Cat est un entier.

donnée utile : l'historique est contenu dans cinq fichiers : Cat1.PRO, Cat2.PRO...

relation : la procédure détermine
- si Systrendu est égal à S ou au plus grand nombre d'éléments de S trouvés dans un fichier Cat...PRO,
- et si Syshisto est égal à un élément d'un fichier Cat...PRO composé de Systrendu et de la suite de TRAITEMENT de cet élément,
- et si Cat est égal au numéro de catégorie qui contient Syshisto.

condition d'application :
in(ground,any,any,any) : out(ground,ground,ground,ground).

- procédure egalsyst(S, L, B).

S, B sont des listes dont les éléments sont de type TRAITEMENT,
L est une liste de listes dont les éléments sont de type TRAITEMENT.

relation : la procédure détermine si B est un élément de L qui est égal ou inclu dans S.

condition d'application :
in(any,ground,any) : out(ground,ground,ground).

- procédure verifa(A, B).

A, B sont des listes dont les éléments sont de type TRAITEMENT.

relation : la relation détermine si A est inclu ou égal à B.

condition d'application :
in(any,ground) : out(ground,ground).
in(ground,any) : out(ground,ground).

1.2. L'identification de la nature

Cette partie a pour objectif de déterminer la nature du système donné par l'utilisateur.

- procédure elecompl(S, Systfich, Systrendu, Etat).

S,Systfich,Systrendu sont des listes dont les éléments sont de type TRAITEMENT,
Etat est un entier.

relation : soit A le nombre de traitements de S,
soit B le nombre de traitements de Systrendu,
soit C le nombre de traitements de Systfich.

la procédure détermine si Etat est égal à
1 lorsque A = B = C,
2 lorsque A = B < C,
3 lorsque A > B = C,
4 lorsque A > B < C.

condition d'application :
in(ground,ground,ground,any) : out(ground,ground,ground,ground).
in(any,any,any,ground) : out(ground,ground,ground,ground).

2. La trace locale

2.1. Création de la trace locale

L'objectif est de construire la trace locale d'un système de peinture donné. Il s'agit de notre connaissance dynamique vue dans un chapitre V.

La trace locale consiste à avoir des informations sur les règles qui se sont éxécutées lors de la recherche des systèmes économiques. Pour ce faire, pour chaque traitement du système de peinture,
- nous écrivons le nom du traitement et l'état courant du support après application du traitement sur un fichier ouvert.
- nous éxécutons les règles suivantes : convient, cherchparam, calculn. En s'exécutant, ces règles vont écrire des informations sur un fichier ouvert.

Les procédures du programme ont été quelque peu modifiées. Ce n'est pas au niveau logique que se situe la modification mais au niveau des effets de bord. Ainsi par exemple, la spécification de la procédure convientnat(Natf, Nats) devient :

- procédure convientnat(Natf, Nats).

Natf, Nats sont des listes dont les éléments sont de type VALEUR.

relation : la procédure détermine :
- si Natf est une liste vide où si Nats est une liste vide,
- ou si la valeur de critère représentée par Natf est également représentée par Nats,
- ou si membre(Nat, Nats) réussit.

condition d'application :
in(any,any) : out(ground,any).

effet de bord : si la règle imprime(1) est dans la base de règle alors la procédure écrit sur un fichier ouvert "son nom", "le numéro de la règle qui réussit ou l'échec de la procédure" et "les données concernant cette procédure".

Il en est de même pour toutes les autres règles. Celles-ci étant déjà spécifiées dans [DEC86], il est inutile de répéter les spécifications.

En raison de ces ajouts, deux types de problèmes sont apparus.

a) la récursivité

Si la procédure est récursive, à chaque passage, la procédure va écrire dans le fichier son identifiant de règle. Par conséquent, dans la trace, pour une exécution de règle récursive, nous aurons N identifants de la règle, car elle a été appelée N fois.

L'identifiant de règle qui nous intéresse est toujours le dernier. En effet quand nous écrivons sur le fichier, nous sommes sûr que la procédure a réussi et donc a fini tous ses traitements.

b) l'instruction "cut"

Si à l'intérieur de la procédure il y a un cut, en cas de réussite il n'y a aucun problème. Par contre, en cas d'échec, la procédure n'exécute pas les instructions écrivant sur le fichier l'échec de la règle. Par conséquent, quand il y a un "cut" dans une procédure, celle-ci est divisée en deux parties, celle avant et celle après le "cut". Il faut prévoir par partie un message d'échec. Ainsi, si la première partie de la procédure avant le "cut" réussi mais que la deuxième échoue, le deuxième message d'échec servira. Ce qui nous entraine à dire qu'il faut éviter le plus possible les cuts car on perd de la lisibilité.

En effet, - quand il n'y a pas de "cut" :

```
procedure P :-  CORPS DE P,
                WRITE(p, reussi).  /* corps a réussi */

          P :-  WRITE(p, echec), fail. /* corps a échoué. */
```

- quand il y a un "cut" :

```
procédure P :-  CORPS A DE P,  !,
                (CORPS B DE P,
                 WRITE(p, reussi);  /* corps A et B ont réussi. */
                 WRITE(p, echec), fail).  /* corps B a échoué.  */

          P :- WRITE(p, echec),fail. /* corps de A a échoué.    */
```

- procédure traceloc(Syst).

Syst est une liste dont les éléments sont de type TRAITEMENT.

Pré : les traitements doivent convenir techniquement.

condition d'application :
in(ground) : out(ground).

effet de bord : le fichier TRACE.PRO contient la trace locale de la liste de TRAITEMENTs Syst.

- procédure tracetrait(Syst,Ef).

Syst est une liste dont les éléments sont de type TRAITEMENT,
Ef est de type CRITERE_DE_DESCRIPTION.

condition d'application :
in(ground,ground) : out(ground,ground).

effet de bord : le fichier TRACE.PRO contient la trace locale de la liste de TRAITEMENTs Syst en partant de l'état final Ef.

2.2. Outils pour l'exploitation de la trace locale

Cette classe de procédures est une sorte de boîte noire. Elle nous permet de prendre des éléments hors de la trace sans connaître la représentation interne de la trace. Trace s'apparente donc à un type abstrait.

- procédure unitetrt(Trt, Trace, Trttrace).

A est de type NOM_DE_REGLE,
Trace, Trttrace sont des suites d'éléments.

Pré : un élément est la représentation interne d'un IDENTIFIANT DE REGLE.

relation : la procédure détermine si Trttrace contient une partie de Trace, correspondant à la trace du traitement.

condition d'application :
in(any,ground,any) : out(ground,ground,gound).

- procédure recherche(A, Trace, Ele).

A est de type NOM_DE_REGLE,
Trace est une liste d'éléments,
Ele est un élément.

Pré : un élément est la représentation interne d'un IDENTIFIANT DE REGLE. Trace représente la trace locale d'un traitement.

relation : la procédure détermine si Ele est un élément de Trace et si le nom de la règle de Ele est A.

condition d'application :
in(any,ground,any) : out(ground,ground,ground).

- procédure etude(A, Trace, Liste).

A est de type NOM_DE_REGLE,
Trace,Liste sont des listes d'éléments,

Pré : un élément est la représentation interne d'un IDENTIFIANT DE REGLE. Trace représente la trace locale d'un traitement.

relation : la procédure détermine si Liste contient
- l'élément de Trace de nom de règle A,
  * contient les éléments des enfants de A ayant réussi lorsque A a réussi,
  * contient les éléments des enfants de A ayant echoué lorsque A a échoué.

condition d'application :
in(ground,ground,any) : out(ground,ground,ground).

- procédure recherchenfant(Enf, Trace, Ele, R).

  Ele, Trace sont des suites d'éléments,
  Enf sont des suites de type NOM_DE_REGLE,
  R est de type VALEUR.

Pré : un élément est la représentation interne d'un IDENTIFIANT DE REGLE. Trace représente la trace locale d'un traitement.

relation : la procédure détermine si Ele contient les éléments de Trace dont les noms de règles sont Enf. Ceux-ci ont :
- réussi lorsque R est différent de la valeur "échec",
- echoué lorsque R est égal à la valeur "échec".

condition d'application :
in(ground,ground,any,any) : out(ground,ground,ground,ground).

- procédure etupar(Regle, Trace, Liste).

  Trace est une liste d'éléments,
  Liste est un élément,
  Regle est de type NOM_DE_REGLE.

Pré : un élément est la représentation interne d'un IDENTIFIANT DE REGLE.
Trace représente la trace locale d'un traitement.

relation : la procédure détermine si Liste contient l'élément de Trace qui est le parent du nom de la règle Regle.

condition d'application :
(in(ground,ground,any) : out(ground,ground,ground).

3. L'explication commerciale

L'objectif est de donner à l'utilisateur des informations économiques sur chaque traitement du système et sur le système lui-même.

Ainsi, il pourra connaître le prix total du système au mètre carré, le temps de séchage, ... Avec ces informations, l'utilisateur comprend pourquoi un système est préféré à un autre.

Les spécifications des procédures concernant cette partie sont dans le chapitre interface. En effet, il n'y a presque pas de traitement, il ne s'agit que de production d'informations à l'écran.

4. L'explication générale du système

L'objectif est de présenter à l'utilisateur en quoi chaque

traitement du système améliore l'état du support et convient au support.

- travail(X).

  X est un entier.

pré : X vaut 4.

effet de bord : si on a étudié un système précédemment, la procédure a pour effet d'expliquer ce système à l'utilisateur avec plus de précisions.

- expli2niv(Syst,Trace).

  Syst est une liste dont les éléments sont de type TRAITEMENT,
  Trace est une liste d'éléments.

pré : Trace est la trace locale du système Syst.

condition d'application :
in(ground,ground) : out(ground,ground).

effet de bord : à partir de Trace et de Syst apparaissent à l'écran les explications générales du système, c'est-à-dire que chaque traitement de ce système fait avancer et convient et nous arrivons à cet état courant.

5. Plus de précisions sur certaines règles

L'objectif est de donner des explications à l'utilisateur sur les parents d'une règle où sur la règle qu'il demande.

- procédure travail(X).

  X est un entier.

pré : X vaut 5.

effet de bord : la procédure sort des phrases à l'écran, afin d'expliciter une règle en profondeur d'un traitement d'un système.

- procédure travail(X).

  X est un entier.

pré : X vaut 6.

effet de bord : la procédure sort des phrases à l'écran afin d'expliciter le parent d'une règle d'un traitement du système.

6. La recherche de l'utilité d'un traitement

L'objectif est de donner à l'utilisateur des explications concernant un traitement par rapport à un problème posé.

- procédure travail(X).

X est un entier.

pré : X vaut 7.

effet de bord : la procédure sort des phrases à l'écran expliquant où se situe un traitement par rapport à un problème posé.

7. L'explication de l'inexistance de solution

L'objectif est de présenter à l'utilisateur les solutions partielles qui par définition ont été abandonnées. Ainsi, il pourra connaître la raison de l'inexistance de systèmes terminaux.

- procédure travail(X).

X est un entier.

pré : X vaut 3.

effet de bord : la procédure sort à l'écran les systèmes de peinture des catégories 4 et 5.

VI.2.5.2. Les procédures de bas niveau

Cette classe de procédures se présente comme une boîte à outils. Elles sont pour la plupart reprises du mémoire de K. Declerq et J. Parache. Par conséquent, il nous semble inutile de répéter les spécifications de ces procédures.

VI.2.5.2.1. Les procédures de manipulation de listes

- procédure tri(A,B,C).

A,B sont des listes d'atomes,
C est un entier.

pré : C appartient à [1,2,4].
si C = 1 alors A est une liste de type VECTEUR GENERAL.
si C = 2 ou 4 alors A est une liste de type VECTEUR D'ETAT COMMERCIAL.

relation : la procédure détermine si B contient A trié sur sa valeur

de critère de nom de critère "eval" selon le critère C.
si C = 1 ou 4, B est ordonné de manière croissante.
si C = 2, B est ordonné de manière décroissante.

condition d'application :
in(ground,any,ground) : out(ground,ground,ground).

- procédure sup(A,B,C).

  A,B sont des atomes.
  C est un entier.

pré : C appartient à [1,2,4].
si C = 1 alors A et B sont de type VECTEUR GENERAL.
si C = 2 ou 4 alors A et B est de type VECTEUR D'ETAT COMMERCIAL.

relation : la procédure détermine
- dans le cas ou C = 2, si A >= B,
- dans le cas ou C = 1 ou 4, si A <= B.

condition d'application :
in(ground,any,ground) : out(ground,ground,ground).
in(var,ground,ground) : out(ground,ground,ground).

- procédure nombre(Y,L).

  Y est une liste d'atomes,
  L est un entier.

relation : la procédure détermine si L est la taille de la liste Y.

condition d'application :
in(ground,any) : out(ground,ground).

- procédure max(L,A).

  L est une liste d'entiers,
  A est un entier.

relation : la procédure détermine si A est l'élément le plus grand de L.

condition d'application :
in(ground,any) : out(ground,ground).

- procédure enlevelele(L,L1,E).

  L est une liste d'atomes,
  L1 est une liste d'atomes,
  E est un atome.

relation : la procédure détermine si E est le dernier atome de L et si L1 est L dont le dernier atome est supprimé.

condition d'application :

in(any,ground,ground) : out(ground,ground,ground).
in(ground,any,any) : out(ground,ground,ground).

- procédure egalprod(L,P,L1).

  L est une liste de listes d'atomes
  P est un atome,
  L1 est une liste d'atomes.

relation : la procédure détermine si L1 contient P et si L contient L1.

condition d'application :
in(ground,ground,any) : out(ground,ground,ground).

VI.2.5.2.2. Manipulation de fichiers

- procédure lire(A).

  A est une liste d'atomes.

pré : il y a un fichier ouvert en lecture.

relation : la procédure détermine si A contient le contenu du fichier ouvert, et si le dernier atome de A est l'élément vide.

condition d'application :
in(var) : out(ground).

- procédure lectfich(Nom,A).

  Nom est un atome,
  A est une liste d'atome.

pré : Nom est un nom de fichier existant.

relation : la procédure détermine si A contient le contenu du fichier de nom Nom, et si le dernier atome de A est l'élément vide.

condition d'application :
in(ground,var) : out(ground,ground).

- procédure écrirefich(L).

  L est une liste d'atomes.

relation : la procédure détermine si le contenu de L est sur le fichier ouvert en écriture.

condition d'application :
in(ground) : out(ground).

effet de bord : le fichier ouvert en écriture contient L.

VI.2.5.2.3. Procédures de manipulation d'un vecteur d'état commercial

- procédure extraction(L,S).

  L,S sont des listes d'atomes,

pré : L est un vecteur général,
S est une liste de valeurs de critères de nom de critère 'sol'.

relation : la procédure détermine si S est égal à chaque valeur de critère de nom de critère 'sol' de la liste L.

condition d'application :
in(ground,any) : out(ground,ground).

- procédure extev(Et,Evalu).

  Et est de type LISTE_DE_CRITERE,
  Evalu est un entier.

relation : la procédure détermine si Evalu est égal à la valeur de critère du critère de description dont le nom de critère est evalu dans le VECTEUR D'ETAT COMMERCIAL dont la représentation interne est Et.

condition d'application :
in(ground,any) : out(ground,ground).

- procédure extprom(Et,Prom).

  Et est de type LISTE_DE_CRITERE,
  Prom est un entier.

relation : la procédure détermine si Prom est égal à la valeur de critère du critère de description dont le nom de critère est prom dans le VECTEUR D'ETAT COMMERCIAL dont la représentation interne est Et.

condition d'application :
in(ground,any) : out(ground,ground).

- procédure extfac(Et,Fac).

  Et est de type LISTE_DE_CRITERE,
  Fac est un atome.

relation : la procédure détermine si Fac est égal à la valeur de critère du critère de description dont le nom de critère est fac dans le VECTEUR D'ETAT COMMERCIAL dont la représentation interne est Et.

condition d'application :
in(ground,any) : out(ground,ground).

- procédure extnpro(Et,Npro).

  Et est de type LISTE_DE_CRITERE,
  Npro est un entier.

relation : la procédure détermine si Npro est égal à la valeur de critère du critère de description dont le nom de critère est npro dans le VECTEUR D'ETAT COMMERCIAL dont la représentation interne est Et.

condition d'application :
in(ground,any) : out(ground,ground).

- procédure extodeur(Et,Od).

  Et est de type LISTE_DE_CRITERE,
  Od est un atome.

relation : la procédure détermine si Od est égal à la valeur de critère du critère de description dont le nom de critère est odeur dans le VECTEUR D'ETAT COMMERCIAL dont la représentation interne est Et.

condition d'application :
in(ground,any) : out(ground,ground).

- procédure extanc(Et,Anc).

  Et est de type LISTE_DE_CRITERE,
  Anc est un atome.

relation : la procédure détermine si Anc est égal à la valeur de critère du critère de description dont le nom de critère est anci dans le VECTEUR D'ETAT COMMERCIAL dont la représentation interne est Et.

condition d'application :
in(ground,any) : out(ground,ground).

- procédure extpis(Et,Pis).

  Et est de type LISTE_DE_CRITERE,
  Pis est un atome.

relation : la procédure détermine si Pis est égal à la valeur de critère du critère de description dont le nom de critère est maplip dans le VECTEUR D'ETAT COMMERCIAL dont la représentation interne est Et.

condition d'application :
in(ground,any) : out(ground,ground).

- procédure extrou(Et,Rou).

  Et est de type LISTE_DE_CRITERE,
  Rou est un atome.

relation : la procédure détermine si Rou est égal à la valeur de critère du critère de description dont le nom de critère est maplir dans le VECTEUR D'ETAT COMMERCIAL dont la représentation interne est Et.

condition d'application :
in(ground,any) : out(ground,ground).

- procédure extprix(Et,Prix).

  Et est de type LISTE_DE_CRITERE,
  Prix est un entier.

relation : la procédure détermine si Prix est égal à la valeur de critère du critère de description dont le nom de critère est pm dans le VECTEUR D'ETAT COMMERCIAL dont la représentation interne est Et.

condition d'application :
in(ground,any) : out(ground,ground).

- procédure exttsech(Et,Tps).

  Et est de type LISTE_DE_CRITERE,
  Tps est un couple.

relation : la procédure détermine si Tps est égal à la valeur de critère du critère de description dont le nom de critère est tsech dans le VECTEUR D'ETAT COMMERCIAL dont la représentation interne est Et.

condition d'application :
in(ground,any) : out(ground,ground).

- procédure extcomp(Et,Comp).

  Et est de type LISTE_DE_CRITERE,
  Comp est un atome.

relation : la procédure détermine si Comp est égal à la valeur de critère du critère de description dont le nom de critère est comp dans le VECTEUR D'ETAT COMMERCIAL dont la représentation interne est Et.

condition d'application :
in(ground,any) : out(ground,ground).

- procédure extqual(Et,Qual).

  Et est de type LISTE_DE_CRITERE,
  Qual est un atome.

relation : la procédure détermine si Qual est égal à la valeur de critère du critère de description dont le nom de critère est qual dans le VECTEUR D'ETAT COMMERCIAL dont la représentation interne est Et.

condition d'application :
in(ground,any) : out(ground,ground).

- procédure extsol(Et,Sol).

  Et est de type LISTE_DE_CRITERE,
  Sol est une liste dont les éléments sont de type TRAITEMENT.

relation : la procédure détermine si Sol est égal à la valeur de critère du critère de description dont le nom de critère est sol dans le VECTEUR D'ETAT COMMERCIAL dont la représentation interne est Et.

condition d'application :
in(ground,any) : out(ground,ground).

VI.2.5.2.4. Procédure de manipulation d'un vecteur d'état

- procédure extitemsvect(Et,Nat,Forme,Env,Ap,Car,Es,As,Res,Pro,Dt).

  Et est de type LISTE_DE_CRITERE,
  Nat,Forme,Env,Ap,Car,Es,As,Res,Pro,Dt sont des atomes.

relation : la procédure détermine si Nat, Forme, Env, Ap, Car, Es, As, Res, Pro, Dt sont égaux aux valeurs de critère de critère de description dont les noms de critère sont respectivement Nature, Forme, Env, Ap, Car, Es, As, Res, Pro, Dt, dans le VECTEUR D'ETAT dont la représentation interne est Et.

condition d'application :
in(ground,any,any,any,any,any,any,any,any,any,any) :
out(ground,ground,ground,ground,ground,ground,ground,ground,ground,gro
und,ground,ground).

VI.2.5.2.5. Procédure de manipulation de la base de faits

- procédure lirequestion(A).

A est une liste dont les éléments sont de type LISTE_DE_CRITERE.

relation : la procédure détermine si A contient la liste de chaque fait 'question'.

condition d'application :
in(any) : out(ground).

VI.2.5.3. L'interface

Les procédures suivantes sortent à l'écran des phrases ou posent des questions à l'utilisateur. Les phrases sont explicitées dans le programme PROLOG. Ainsi, chaque spécification de procédure fait référence à son code dans le programme.

Les spécifications des procédures SORTPHR, RENVLISTE, SORTLISTE ont été spécifiées dans [DEC86].

- procédure menu.

effet de bord : la procédure sort à l'écran le menu du programme.

- procédure sortieprepa(Pre).

Pre est une liste dont les éléments sont de type TRAITEMENT.

effet de bord : la procédure sort à l'écran la liste Pre.

- procédure sortie(L).

L est une liste de listes dont les éléments sont de type TRAITEMENT.

effet de bord : la procédure sort à l'écran chaque élément de la liste L espacé de la phrase "la suite de traitement à appliquer est :".

- procédure sortie1niv(L).

L est de type PHRASES.

effet de bord : la procédure sort à l'écran les phrases contenues dans L.

- procédure expl2com(Sys).

Sys est une liste dont les éléments sont de type TRAITEMENT.

Pré : Sys doit être un système techniquement correct.

effet de bord : la procédure sort à l'écran l'explication du vecteur commercial terminal de Sys.

- procédure impreseffetdebord(Trt, A).

  A est une liste dont les éléments sont de type VALEUR DE CRITERE,
  Trt est de type TRAITEMENT.

effet de bord : la procédure sort à l'écran l'explication des caractéristiques économiques A du traitement Trt.

- procédure sortiefam(L).

  L est une liste d'éléments.

Pré : un élément est de type IDENTIFIANT D'UNE REGLE.

effet de bord : la procédure sort à l'écran les phrases correspondant à L. La première phrase est suivie de "car".

- procédure sortieenf(L).

  L est une liste d'éléments.

Pré : un élément est de type IDENTIFIANT D'UNE REGLE.

effet de bord : la procédure sort à l'écran les phrases correspondant à chaque élément de L.

- procédure sortiepar(L).

  L est un élément.

Pré : un élément est de type IDENTIFIANT D'UNE REGLE.

effet de bord : la procédure sort à l'écran la phrase "le parent de la règle est" et ensuite les phrases correspondant à L.

- procédure présentationetat(Et).

  Et est est un LISTE_DE_CRITEREs.

effet de bord : la procédure sort à l'écran des phrases correspondant à l'explication du VECTEUR D'ETAT.

- procédure menucom(Rep).

  Rep est un caractère.

condition d'application :
in(var) : out(ground).

effet de bord : la procédure pose à partir de l'écran une question à l'utilisateur et la réponse est placée dans Rep.

- procédure questionregletrt(Regle,Trt).

Regle est un entier,
Trt est de type TRAITEMENT.

condition d'application :
in(var,var) : out(ground,ground).

effet de bord : la procédure pose deux questions à l'utilisateur, celui-ci,à partir du clavier, entre un numéro de règle Regle et un traitement Trt.

- procédure poser(Phr,Rep).

Phr est de type PHRASE,
Rep est un caractère.

condition d'application :
in(ground,var) : out(ground,ground).

effet de bord : la procédure pose une question Phr à l'utilisateur, celui-ci, à partir du clavier, entre sa réponse Rep.

VI.2.6. Les aspects non implémentés

Cette section a pour objet de décrire les aspects qui n'ont pu être implémentés faute de temps. Il s'agit d'aspects ne nécessitant que très peu d'analyse mais qui cependant sont indispensables pour compléter l'environnement du système expert développé.

- Explication "glossaire"

Lorsque le système expert justifie le choix ou le rejet d'un système expert, il utilise des concepts propres à la peinture ou à la stratégie qu'il a employé. Ces concepts peuvent être inconnus ou trop peu explicites pour l'utilisateur. Il doit donc avoir la possibilité de pouvoir signaler au système qu'il désire de plus amples détails sur les termes qu'il ne connaît pas.

Exemples : * liant
* solvant
* coefficient d'importance
.
.
.

- Fiches techniques et principes généraux du domaine de la peinture

D'une part, le peintre professionnel peut être intéressé par les caractéristiques techniques détaillées d'un nouveau traitement qui sont accessibles dans des fiches techniques. D'autre part,

l'utilisateur naïf peut être désireux de connaître de manière tout à fait abstraite quels sont les grands principes appliqués dans le choix d'une peinture.

- Les fonctions supplémentaires

Les systèmes de peinture candidats au problème posé par l'utilisateur peuvent présenter des caractéristiques supplémentaires par rapport aux exigences de ce dernier. Ces caractéristiques ont été nommées au point IV.2.1.2. les "fonctions supplémentaires". Celles-ci reprennent également les caractéristiques quasi obligatoires qu'un système de peinture doit respecter pour être de bonne qualité. Pour les supports métalliques, il est nécessaire de prévoir une protection contre la corrosion et pour les boiseries, une protection contre l'humidité est indispensable. Les fonctions supplémentaires devraient intervenir dans l'évaluation du système considéré.

- Les incompatibilités

L'absence de solution peut être provoqué par une incompatibilité au niveau des exigences de l'utilisateur. Il devrait être possible d'établir une liste statique d'incompatibilités ou de déterminer selon certaines règles, si la description des états du support exposée par l'utilisateur comporte des incompatibilités qui empêchent le système de trouver une solution à ce problème.

- La phase d'initialisation

Le système expert que nous avons élaboré doit comporter une phase d'initialisation. Elle permet de fixer quels sont les produits actuellement en stock, quels sont les coefficients d'importance et de pondération en fonction du type de clientèle et quel est le type de client qui utilisera le système expert. Grâce à cette phase la base de données peut être mise à jour en ce qui concerne ces connaissances.

VII. CONCLUSION

Le travail effectué au cours de ce mémoire a permis de prendre en considération les critères commerciaux dans la construction d'un système de peinture. La recherche d'un tel système se fait non plus selon une méthode exhaustive mais bien heuristique. Cette méthode offre l'avantage de ne développer que les solutions les plus prometteuses.

Le système expert élaboré est également capable de fournir la justification des résultats qu'il donne et le raisonnement sous-jacent. Un tel processus d'explication n'a guère pu être inspiré d'une démarche classique. En effet, un problème similaire à celui que nous avons abordé n'est pas traité dans la littérature dont nous disposions. Le processus d'explication afférent au conseil en peinture a été construit de toute pièce.

Le système Conseil comporte finalement deux objectifs : réaliser une recherche dans le domaine de l'intelligence artificielle et développer un outil de conseil pour la mise en peinture.

La partie intelligence artificielle du travail implique certaines propriétés du système.

1. Son élaboration doit permettre de rendre la technologie informatique plus naturelle et plus acceptable aux experts.

2. Il doit comprendre un outil de consultation des données du système de façon à ce que les utilisateurs puissent connaître les connaissances relatives au domaine d'expertise. Il doit également être facilement extensible, de nouvelles connaissances doivent pouvoir être ajoutées à la base de connaissances.

3. Il doit être basé sur un modèle général de recherche, indépendamment du domaine d'expertise avec l'espoir que cette structure générale puisse être appliquée pour d'autres problèmes comportant des actions similaires; le raisonnement que le système expert suit devrait pouvoir être repris, moyennant quelques modifications, pour résoudre un problème semblable.

4. Il doit être habillé d'une interface rapide et conviviale. Elle doit permettre de donner accès à un système d'intelligence artificielle, pour des utilisateurs qui n'ont pas d'expérience avec l'intelligence artificielle ou avec les ordinateurs en général et qui peuvent n'avoir que très peu de connaissances sur le domaine d'expertise.

5. Il doit inclure des techniques vérifiant la complétude et la cohérence de la base de connaissances.

Nous allons voir ce qu'il en est pour le système expert que nous avons développé.

Nous pouvons espérer que le système Conseil atténue et même fasse

disparaître l'aversion de certains chimistes envers les capacités de ce système et plus généralement envers l'informatique. Une utilisation agréable du système leur permettra d'évaluer le degré de difficulté des problèmes qu'il peut résoudre.

L'informatisation que nous avons réalisée pour le conseil en peinture a non seulement l'avantage de donner toujours la même suite de solutions pour un même problème, mais de plus, l'ensemble des produits est très facilement extensible. Tout nouveau produit peut sans aucune difficulté être ajouté à la base de connaissances, à la condition bien sûr d'utiliser le même formalisme que pour les traitements déjà existants et de spécifier toutes ses caractéristiques. Le système de recherche considérera ce nouveau produit tout comme les autres. Nous en avons d'ailleurs fait l'expérience avec l'incorporation des traitements appropriés pour le bois. Les adaptations qui ont dû être apportées trouvent leur source dans l'ajout d'une nature de support, ce qui implique la prise en considération de quelques éléments spécifiques à cette nature. Mais lorsque tous les types de supports seront intégrés, l'addition d'un produit ne demande aucune modification. L'adjonction des critères commerciaux et du processus d'explication n'entrave en aucune manière cette facilité.

Un autre avantage que présente le programme Conseil est que la modélisation du système de recherche et du processus d'explication peut être réutilisée pour tout autre domaine d'application impliquant les mêmes étapes : un état de départ, un état final et une suite d'opérateurs pour passer de l'un à l'autre. Les deux états constituent un ensemble de caractéristiques de nature identique afin de pouvoir les comparer, et les opérateurs sont soumis à des conditions de choix régies par une série de critères. Le processus d'explication ne s'inspirant pas des méthodes classiques offre une nouvelle alternative pour adjoindre des explications aux décisions d'un système expert.

Les autres propriétés requises par l'intelligence artificielle font l'objet d'extensions qui semblent être une suite logique à notre travail.

VII.1. EXTENSIONS

La première extension à apporter, et qui de loin est la plus urgente est de concevoir une interface rapide et conviviale. L'utilisateur doit n'éprouver aucune difficulté à formuler les questions qu'il désire poser et à comprendre les réponses fournies par le système. Celles-ci doivent être dépourvues de toute notation propre à l'informatique. Il doit également pouvoir accéder aisément à toutes les informations qu'il est autorisé à consulter. Un refus doit lui être présenté de façon à ne pas le frustrer. En outre, la terminologie employée doit s'adapter à chaque type d'utilisateur.

Au stade actuel du programme Conseil, les questions sont posées par l'intermédiaire d'un menu et la terminologie utilisée dans les

réponses est la même quel que soit le type de l'utilisateur. L'interface idéale pour un tel système devrait comprendre un outil de compréhension et de génération de langage naturel. La génération s'avère nécessaire lors de la phase d'explication : de chaque règle devrait pouvoir être généré automatiquement un texte en langue naturelle explicitant la règle considérée. Une telle génération suppose une modélisation des règles.

Cette partie du système Conseil dépasse le cadre de notre mémoire. En outre, les outils dont nous disposions n'étaient pas du tout adéquats pour réaliser ce travail. En effet, le système Prolog utilisé ne comporte aucune fonction d'interface et de gestion d'écran.

Le deuxième sens dans lequel il est nécessaire d'étendre le système est de concevoir un outil d'acquisition de connaissances utile à une phase qui nous semble très intéressante : la phase de validation caractérisée par des tests effectués par les experts afin de vérifier si les connaissances qu'ils nous ont fournies ont été correctement interprétées et si elles sont complètes. Cette phase implique une collaboration très étroite entre informaticiens, chimistes et commerciaux, les experts vérifiant l'exactitude des règles appliquées (grâce notamment au processus d'explication), les informaticiens apportant les modifications suggérées par les experts. Cette phase de tests peut dans une certaine mesure se passer d'informaticiens. En effet, l'outil d'acquisition de connaissances permettrait aux chimistes et aux commerciaux d'introduire dans le système les nouvelles règles qu'ils jugent utile. Grâce à cet outil, il serait également possible de faire face au cours du temps, à tout changement dans les principes de décision et à l'apparition de nouveaux produits sans devoir faire intervenir un informaticien.

Une troisième extension est de traiter les supports autres que les métaux et les boiseries. Les autres types du support à intégrer sont les maçonneries, les matières plastiques, le verre, la faïence, etc. Comme pour l'extension aux supports en bois, le processus de traitement des supports d'autre nature est le même que celui appliqué aux métaux. Il suffit de le compléter de certaines adaptations dues aux spécificités des autres natures, d'étendre la terminologie et les jeux de valeurs pour les caractéristiques du support.

VII.2. LES PROBLEMES RENCONTRES

Les problèmes auxquels nous avons dû faire face sont de deux ordres : les problèmes humains et les problèmes techniques.

1. Les problèmes humains

- Les experts éprouvent de nombreuses difficultés à exprimer comment ils prennent leurs décisions dans le domaine d'application que nous avons considéré, et quels sont les fondements de ces décisions. Ces difficultés sont d'autant plus grandes lorsqu'ils ont acquis une

expérience qui leur confère un certain automatisme dans leurs actes. Les modèles qu'ils nous confient sont plutôt descriptifs. Leurs conclusions semblent être basées sur des mécanismes mal définis indiquant un manque de connaissances formelles concernant les relations entre toutes les variables du problème à résoudre. Ils raisonnent par cas particulier, de manière exhaustive, plutôt que par application de principes généraux. Les experts connaissent les décisions à prendre mais ont du mal à communiquer le pourquoi de celles-ci et le raisonnement qu'ils ont suivi.

C'est à l'issue de nombreuses questions et de beaucoup de temps que nous avons pu déceler les principes qu'inconsciemment ils appliquent.

- Chaque expert ayant acquis une expérience personnelle, nous nous sommes heurtés à de nombreuses contradictions quant à leurs avis. En effet, leur expérience leur ont apporté certaines préférences dans la gamme de produits, ce qui peut impliquer des décisions différentes face à un même problème de peinture. Il a cependant fallu que nous tranchions bien qu'au départ nous ignorions tout de la peinture. Les degrés de croyance que nous avons accordés aux révélations des experts et le contenu de la base de faits sont donc éventuellement à revoir.

- Le manque de confiance que les experts accordent aux capacités de l'informatique et des ordinateurs les empêchent de nous communiquer certaines connaissances qu'ils détiennent, en particulier celles qui sont relatives au "savoir-faire" et à l'intuition, par crainte que ces informations ne puissent être interprétées de manière correcte.

  Leur réticence vis-à-vis du caractère réalisable de notre projet (seuls quelques chimistes croient aux capacités réelles du programme Conseil) pose des problèmes au cours des entretiens que nous avons eus avec eux. Il est donc nécessaire de créer un climat de confiance dès le départ et de bien exposer le but de notre travail : récolter des informations afin d'aider les clients à résoudre des problèmes classiques de peinture et non pas remplacer le travail des experts par un ordinateur qui leur dicterait les décisions à prendre. Il n'est pas toujours très aisé de faire accepter cette idée.

Ces problèmes rendent d'autant plus nécessaire la phase des tests.

2. Les problèmes techniques

Les problèmes techniques auxquels nous avons été confrontés sont dus à la faible puissance de la machine (HP 150) sur laquelle nous avons élaboré le programme Conseil.

- Les problèmes d'interface ont déjà été abordés.

- La capacité de la mémoire centrale s'est très rapidement révélée

trop petite pour pouvoir exécuter le programme dans sa totalité. Nous avons été contraints de travailler sur une toute petite partie de la base de connaissances qui à l'origine contenait l'ensemble des produits pour métaux et pour bois de la gamme Trimetal. Il nous a également été impossible d'intégrer tous les modules de programmation; ces derniers ont dû être testés de façon plus ou moins autonome. Il fallait choisir entre un programme peu gourmand en place mémoire et illisible, et un programme clair mais plus gourmand. Comme dans [DEC86], nous avons opté pour la deuxième solution, en vue d'extensions éventuelles qui pourraient être réalisées par d'autres que nous. Ce manque de place mémoire a occasionné de grandes pertes de temps.

- Le temps d'exécution du programme est assez long. Les performances sont très peu satisfaisantes.

L'utilisation du langage est assez coûteuse au point de vue de la machine. En effet, le nombre de traitements de listes impliqués par la reconnaissance de ce qu'une liste est l'instance d'une autre est fonction de la longueur et du nombre de listes imbriquées, et du nombre de variables dans le pattern. Le mécanisme de backtracking est également coûteux, car il suppose une gestion de pointeurs assez complexe : tous les objectifs qui sont en cours de preuve reçoivent un pointeur, de façon à ce que l'interpréteur puisse les retrouver aisément au moment où il doit "défaire" les instanciations déjà établies. Il est certain que l'utilisation d'un interpréteur PROLOG n'est pas l'idéal au point de vue des performances.

Nous pouvons également expliquer les mauvaises performances de notre prototype au niveau de la vitesse d'exécution par le caractère verbeux du texte du programme. Ce caractère résulte du choix que nous avons opéré pour faciliter la lisibilité, la modifiabilité et les tests de la base de connaissances. Nous aurions pu, par exemple, supprimer le nom de toutes les caractéristiques des traitements, puisque seules leurs valeurs sont nécessaires. Chaque caractéristiques auraient une place fixe dans la liste qui les contient. Cet ordre permet de distinguer le type de chaque élément.

Le gain de performance réalisé grâce à une recherche avec fonction d'évaluation au lieu d'une recherche en profondeur d'abord (selon laquelle toutes les solutions techniquement correctes sont générées avant de pouvoir choisir la plus intéressante commercialement parlant) a été compensé par l'ajout des traitements pour bois, venant grossir la base de faits.

Il est clair qu'un système expert doit par définition proposer des solutions correctes mais également les proposer dans des temps de réponses raisonnables. Cet obstacle a occasionné de grandes pertes de temps lors de la phase de tests.

Ces deux types de problèmes devraient être en grande partie résolus lorsque le programme Conseil sera, comme prévu, implémenté sur un ordinateur plus puissant.

BIBLIOGRAPHIE

IJCAI-79 PROCEEDING OF THE SIXTH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE,
Tokiyo, August 20-23, 1979.

IJCAI-81 PROCEEDING OF THE SEVENTH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE,
University of British Columbia Vancouver BC, Canada, August 24-28, 1981.

[BON84] BONNEL A. : L'Intelligence Artificielle Promesses et Réalités
(Inter Editions, Paris), 1984.

[BRA79] BRATKO I. : Implementing Search Heuristics Using the AL1 Advice-Taking System
in IJCAI-79, tome 1, 1979.

[BRO79] BROOKS R. and HEISER J. : Controling Question Asking in a Medical Expert System
in IJCAI-79, tome 1, 1979.

[BUC84] BUCHANAN B.G. and SHORTLIFFE E.H. : Rule-Based Expert System - The MYCIN Experiments of the Stanford Heuristic Programming Project
(Addison - Wesley Publishing Company), 1984, p. 209-292, 331-388, 599-665.

[CLA81] CLANCEY W.J. and LETSINGER R. : Neomycin : Reconfiguring a Rule-Based Expert System for Application to Teaching
in IJCAI-81, tome 2, 1981.

[CLA83] CLANCEY W.J. : The Epistemology of a Rule Based Expert System : a Framework for Explanation
in Artificial Intelligence 20(3), 1983, p. 215-251.

[CLA84] CLARK K.L. and MAC CABE F.G. : Micro-Prolog : Programming in Logic
(Prentice-Hall International Series in Computer Science), 1984.

[CLA87] CLANCEY W.J. : Intelligent Tutoring Systems : a Tutorial Survey
in Current Trends in Expert Systems, A. van Lamsweerde and P. Dufour (Eds.), (Academic Press), 1987, p. 39-77.

[CLO84] CLOCKSIN W.F. and MELLISH C.S. : Programming in Prolog,
(Springer-Verslag, Berlier Heidelberg), seconde édition, 1984.

[CUL81] CULLINGFORD and KRUEGER M.W. and SELFRIDGE M. and BIENKOWSKI M.A. : Towards Automating Explanations
in IJCAI 81, tome 1, 1981.

[DAH79] DAHL V. : Quantification in a Three-Valued Logic for Naturel Language Question-Answering Systems
in IJCAI-79, tome 1, 1979.

[DAV87] DAVIS R. and LENAT D.B. : Knowledge-Based System in Artificial Intelligence,
(Mac Graw Hill International Book Company), 1987, p.35-60, 205-218, 259-408.

[DEC86] DECLERCQ K. et PARACHE J. : Etude et Développement d'un Système Expert Conseil pour la Mise en Peinture,
(Facultés Universitaires Notre-Dame de la Paix, Namur-Belgique), mémoire 1986.

[DEV87] DEVILLE Y. : A Methodology for Logic Program Construction,
(Facultés Universitaires Notre-Dame de la Paix, Namur-Belgique), Thesis 1987.

[DUD81] DUDA R.O. and GASHNIG J.G. : Knowledge-Based Expert System Come of Age
(Byte Publications Inc.), 1981.

[ERM] ERMAN L.D., HAYES-ROTH F., LESSER V.R. and REDDY D.R. : The Hearsay-II Speech-Understanding System : Integrating Knowledge to Resolve Uncertainty
in Expert Systems and AI Applications.

[FEI63] FEIGENBAUM E.A. and FELDMAN J. : Computer and Thought
(Mac Graw Hill), 1963.

[FOX81] FOX M.S. : Reasoning With Incomplete Knowledge in a Resource-Limited Environment : Integrating Reasoning and Knowledge Acquisition
in IJCAI-81, tome 1, 1981.

[GAS79] GASHNIG J. : A Problem Similarity Approach to Devising Heuristics : First Results
in IJCAI-81, tome 1, 1981.

[GRI39] GRIES D. : The Science of Programming
(Library of Congres Cataloging in Publication Data Gries David), 1939.

[HAA81] HAAS A. : Reasoning about Deductive with Unknown Constants
in IJCAI-81, tome 1, 1981.

[HAG81] HAGERT G. : Deductive Modeling of Human Condition
in IJCAI-81, tome 1, 1981.

[HAR81] HARTLEY R.T. : How Expert Should an Expert System Be ?
in IJCAI-81, tome 2, 1981.

[HAY83] HAYES-ROTH F. : The Knowledge-Based Expert Systems : a Tutorial
in COMPUTER, pp 11 - 28, (IEEE), 1984.

[HOL79] HOLLANDER C.R. and REINSTEIN H.C. : A Knowledge-Based Application Definition System
in IJCAI-79, tome 1, 1979.

[HOG84] HOGGER C.J. : Introduction to Logic Programming,
(Academic Press Data Processing No. 21), 1984.

[HOW86] HOWITZ E., HECKERMAN D., NATHWANI B. and FAGAN L. : The Use of a Heuristic Problem-Solving Hierarchy to Facilitate The Explanation of Hypothesis-Directed Reasoning
(Medical Computer Science Group, Stanford University School of Medicine), 1986.

[JAN79] JANAS J.M. : How To Not Say "nil" - Improving Answers to Failing Queries in Data Base Systems
in IJCAI-79, tome 1, 1979.

[KAN81] KANAL L. and KUMAR V. : Branch and Bound Formulation for Sequential and Parallel Game Tree Searching : Preleminary Results
in IJCAI-81, tome 1, 1981.

[KLA79] KLAHR P. : Conditional Answers in Question - Answering Systems
in IJCAI-79, tome 1, 1979.

[JOS85] JOSEPH R.L., ENDSOR J.R., DICKINSON A. and BLUMENTHAL R.L. : An Explanation Facility for an Object-Based Expert System
(CMU-CS), 1985.

[LAS81] LASZLO M. : Some Remarks on Heuristic Search Algorithms
in IJCAI-81, tome 1, 1981.

[LAU81] LAUBSCH J. and EISENSTAD M. : Domain Specific Debugging Aids for Novice Programmers
in IJCAI-81, tome 2, 1981.

[LEB81] LEBOWITZ M. : The Nature of Generalization in Understanding,
in IJCAI 81, tome 1, 1981.

[NIL71] NILSSON N.J. : Problem-Solving Methods in Artificial Intelligence
(Mac Graw-Hill), 1971.

[ROB68] ROBINSON J.A. : The Generlized Resolution Principle,
in Machine Intelligence 3, D. Mitchie Editor,
(Edimburgh University Press), 1968, p.369-426.

[STO79] STOLFO S.J. and HARRISON M. : Automatic Discovery of Heuristics for Nondeterministic
in IJCAI-79, tome 2, 1979.

[SWA81] SWARTOUT W.R. : Explaining and Justifying Expert Consulting Programs

in IJCAI-81, tome 2, 1981.

[WAR83] WARNER HASLING D., CLANCEY W.J. and RENNELS G. : Strategic Explanations for a Diagnostic Consultation System (Department of Computer Science, Stanford University), 1983.

[WEB83] WEBBER B.L. and NILSSON N.J. : Reading in Artificial Intelligence, (Tioga Publishing Company Palo Alto), 1983.

Bibliographie technique

Guide Trimetal : Trimetal Group, (Trimetal-Belgique).

Guide Trimetal : Peintures Faciles, (Trimetal-Belgique).

Guide Trimetal : Digest de la Peinture, (Trimetal-Belgique).

Guide De Keyn : Peindre : Pourquoi pas vous ?, (De Keyn).

Travaux de peinture sur bois.

CONTRIBUTION AU DEVELOPPEMENT

D'UN SYSTEME EXPERT CONSEIL

EN MISE EN PEINTURE

Annexes.

Mémoire présenté en vue de l'obtention du titre de licencié et maître en informatique

Promoteur : Monsieur A. van Lamsweerde

Année académique 1986 - 1987

```
/*            A. BASE DE FAITS                                    */

/* LA RECHERCHE DES SYSTEMES DE PEINTURE.                         */

/* PREMIERE CLASSE DE FAITS.                                      */

/*                  POSTCONDITIONS                                */

post( degraissage, [[nature, [acier, alu, galva, zinc]],
                    [carspec, [prepa]],
                    [etatsurf,[corrosion_non,graisse_non,
                               poussiere_non, proprete_oui,
                               accrochage_oui, nettete_nu]]
                    ]).

post( degraissage, [[nature, [metaux]],
                    [carspec, [prepa]],
                    [etatsurf,[corrosion_non, graisse_non,
                               poussiere_non, proprete_oui,
                               accrochage_oui, nettete_bon_etat]]
                  ]):-
                  etat_de_depart(Ed),
                  valeur_item( [etatsurf, Es], Ed, 1),
                  inclus( [nettete_bon_etat], Es) .


post( lavage_eau_claire, [[nature,[acier]],
                          [carspec, [prepa]],
                          [etatsurf, [corrosion_non,graisse_oui,
                                      poussiere_non,proprete_oui,
                                      accrochage_oui, nettete_nu]]
                                ]) :-
                   etat_de_depart( Ed),
                   valeur_item( [etatsurf, Es], Ed, 1),
                   not(appartient(corrosion_oui, Es)).

post( lavage_eau_claire, [[nature,[galva, zinc]],
                          [carspec, [prepa]],
                          [etatsurf, [corrosion_non,graisse_oui,
                                      poussiere_non, proprete_oui,
                                      Accrochage_oui, nettete_nu]]
                                ]):-
                  etat_de_depart( Ed),
                  valeur_item([etatsurf, Eds], Ed, 1),
                  appartient(nettete_nu, Eds),
                  etat_final( Ef),
                  valeur_item([etatsurf, [recouvrable]], Ef, 1).
```

```
post( lavage_eau_claire, [[nature,[galva]],
                          [carspec, [prepa]],
                          [etatsurf, [corrosion_non,graisse_oui,
                                      poussiere_non, proprete_oui,
                                      Accrochage_A, nettete_nu]]
                         ]):-
                    etat_de_depart( Ed),
                    valeur_item([etatsurf, Eds], Ed, 1),
                    appartient(nettete_nu, Eds),
                    etat_final( Ef),
                    valeur_item([etatsurf, Es], Ef, 1),
                    croiss(accro, A),
                    appartient(Accrochage_A, A),
                    appartient(Accrochage_A, Es).

post( lavage_eau_claire, [[nature,[zinc]],
                          [carspec, [prepa]],
                          [etatsurf, [corrosion_non,graisse_oui,
                                      poussiere_non, proprete_oui,
                                      accrochage_oui, nettete_nu]]
                         ]):-
                    etat_de_depart( Ed),
                    valeur_item([etatsurf, Eds], Ed, 1),
                    appartient(nettete_nu, Eds),
                    etat_final( Ef),
                    valeur_item([etatsurf, Es], Ef, 1),
                    not(appartient( corrosion_oui, Es)).

post( lavage_eau_claire, [[nature,[metaux]],
                          [carspec, [prepa]],
                          [etatsurf, [corrosion_non, graisse_oui,
                                      poussiere_non, proprete_oui,
                                      Accrochage_A, nettete_bon_etat]]
                         ]):-
                    etat_de_depart( Ed),
                    valeur_item( [etatsurf, Es], Ed, 1),
                    inclus([nettete_bon_etat, proprete_non], Es),
                    croiss(accro, A),
                    appartient(Accrochage_A, A),
                    appartient(Accrochage_A, Es) .


post( polissage, [[nature,[cuivre]],
                  [carspec, [prepa]],
                  [etatsurf, [corrosion_non, graisse_non,
                              poussiere_non, proprete_oui,
                              accrochage_oui, nettete_nu]]
                 ]).

post( mordancage, [[nature,[galva, zinc]],
                   [carspec, [prepa]],
                   [etatsurf, [corrosion_non, graisse_oui,
                               poussiere_non, proprete_non,
                               accrochage_oui, nettete_nu]]
                  ]):-
```

```
                    etat_de_depart( Ed),
                    valeur_item([etatsurf, Es], Ed, 1),
                    inclus( [accrochage_non, corrosion_non], Es).

post( lavage_eau_savon, [[nature, [zinc]],
                         [carspec, [prepa]],
                         [etatsurf, [corrosion_non, graisse_oui,
                                     poussiere_non, proprete_oui,
                                     accrochage_oui, nettete_nu]]
                        ]):-
                    etat_de_depart(Ed),
                    valeur_item( [etatsurf, Es], Ed, 1),
                    appartient( corrosion_oui, Es).

post( epoussetage, [[nature,[alu]],
                    [carspec, [prepa]],
                    [etatsurf, [corrosion_oui,graisse_oui,
                                poussiere_non,proprete_oui,
                                accrochage_douteux,nettete_nu]]
                    ]):-
                    etat_de_depart(Ed),
                    valeur_item([etatsurf, Es], Ed, 1),
                    inclus([corrosion_oui], Es).

post( epoussetage, [[nature,[alu]],
                    [carspec, [prepa]],
                    [etatsurf, [corrosion_non, graisse_oui,
                                poussiere_non, proprete_oui,
                                accrochage_douteux, nettete_nu]]
                    ]).

post( epoussetage, [[nature,[acier]],
                    [carspec, [prepa]],
                    [etatsurf, [corrosion_non, graisse_oui,
                                poussiere_non, proprete_oui,
                                accrochage_oui, nettete_nu]]
                    ]).


post( epoussetage, [[nature,[metaux]],
                    [carspec, [prepa]],
                    [etatsurf, [corrosion_non, graisse_oui,
                                poussiere_non, proprete_oui,
                                accrochage_oui, nettete_bon_etat]]
                    ]).

post( epoussetage, [[nature,[galva, zinc]],
                    [carspec, [prepa]],
                    [etatsurf, [corrosion_non, graisse_oui,
                                poussiere_non, proprete_oui,
                                accrochage_oui, nettete_nu]]
                    ]).

post( derouillage, [[nature,[acier,galva]],
                    [carspec, [prepa]],
```

```
                    [etatsurf, [corrosion_non,graisse_oui,
                                poussiere_oui, proprete_oui,
                                accrochage_oui,nettete_nu]]
                   ]):-
                   etat_de_depart(Ed),
                   valeur_item([etatsurf, Es], Ed, 1),
                   appartient(corrosion_oui, Es).


post( poncage, [[nature,[metaux]],
                [carspec, [prepa]],
                [etatsurf, [corrosion_non,graisse_oui,
                            poussiere_oui, proprete_oui,
                            accrochage_oui, nettete_bon_etat]]
               ]):-
                   etat_de_depart(Ed),
                   valeur_item([etatsurf, Es], Ed, 1),
                   inclus([accrochage_non,nettete_bon_etat], Es).


post(appel_specialiste ,[
                [nature, [bois]],
                [carspec, [prepa]],
                [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,Biologique_Bi ,Bleuissement_Bl
,Cloquage_Cl  ,Defaut_De  ,Humidite_Hu  ,Arete_Ar   ,Proprete_Pr
,Poussiere_Po ,Graisse_Gr]]]) :-
                   etat_de_depart(Ed),
                   valeur_item([etatsurf,Es],Ed,1),
                   avanc(Es),
                   (appartient(biologique_fort,Es),
                   concat([Biologique_Bi],[biologique_non],[]);
                   not(appartient(biologique_fort,Es)),
                   croiss(bio,Bi),
                   appartient(Biologique_Bi,Bi),
                   appartient(Biologique_Bi,Es)),
                   croiss(bleu,Bl),
                   appartient(Bleuissement_Bl,Bl),
                   appartient(Bleuissement_Bl,Es),
                   croiss(cloq,Cl),
                   appartient(Cloquage_Cl,Cl),
                   appartient(Cloquage_Cl,Es),
                   croiss(def,De),
                   appartient(Defaut_De,De),
                   appartient(Defaut_De,Es),
                   croiss(humi,Hu),
                   appartient(Humidite_Hu,Hu),
                   appartient(Humidite_Hu,Es),
                   croiss(aret,Ar),
                   appartient(Arete_Ar,Ar),
                   appartient(Arete_Ar,Es),
                   croiss(pouss,Po),
                   appartient(Poussiere_Po,Po),
                   appartient(Poussiere_Po,Es),
                   croiss(prop,Pr),
```

```
                    appartient(Proprete_Pr,Pr),
                    appartient(Proprete_Pr,Es),
                    croiss(graisse,Gr),
                    appartient(Graisse_Gr,Gr),
                    appartient(Graisse_Gr,Es).
avanc(Es) :- appartient(noeud_oui,Es),!.
avanc(Es) :- appartient(excudation_oui,Es),!.
avanc(Es) :- appartient(porosite_oui,Es),!.
avanc(Es) :- appartient(fissuration_oui,Es),!.
avanc(Es) :- appartient(biologique_fort,Es),!.

post(decapage ,[
                    [nature, [bois]],
                    [carspec, [prepa]],
                    [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,Biologique_Bi ,Bleuissement_Bl
,cloquage_non  ,Defaut_De  ,Humidite_Hu  ,Arete_Ar   ,Proprete_Pr
,Poussiere_Po ,Graisse_Gr]]]) :-
                    etat_de_depart(Ed),
                    valeur_item([etatsurf,Es],Ed,1),
                    appartient(cloquage_fort,Es),
                    (appartient(biologique_fort,Es),
                    concat([Biologique_Bi],[biologique_non],[]);
                    not(appartient(biologique_fort,Es)),
                    croiss(bio,Bi),
                    appartient(Biologique_Bi,Bi),
                    appartient(Biologique_Bi,Es)),
                    croiss(bleu,Bl),
                    appartient(Bleuissement_Bl,Bl),
                    appartient(Bleuissement_Bl,Es),
                    croiss(def,De),
                    appartient(Defaut_De,De),
                    appartient(Defaut_De,Es),
                    croiss(humi,Hu),
                    appartient(Humidite_Hu,Hu),
                    appartient(Humidite_Hu,Es),
                    croiss(aret,Ar),
                    appartient(Arete_Ar,Ar),
                    appartient(Arete_Ar,Es),
                    croiss(pouss,Po),
                    appartient(Poussiere_Po,Po),
                    appartient(Poussiere_Po,Es),
                    croiss(prop,Pr),
                    appartient(Proprete_Pr,Pr),
                    appartient(Proprete_Pr,Es),
                    croiss(graisse,Gr),
                    appartient(Graisse_Gr,Gr),
                    appartient(Graisse_Gr,Es).
post(grattage ,[
                    [nature, [bois]],
                    [carspec, [prepa]],
                    [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,Biologique_Bi ,Bleuissement_Bl
,cloquage_non  ,Defaut_De  ,Humidite_Hu  ,Arete_Ar   ,Proprete_Pr
,Poussiere_Po ,Graisse_Gr]]]) :-
```

```
                    etat_de_depart(Ed),
                    valeur_item([etatsurf,Es],Ed,1),
                    appartient(cloquage_faible,Es),
                    (appartient(biologique_fort,Es),
                    concat([Biologique_Bi],[biologique_non],[]);
                    not(appartient(biologique_fort,Es)),
                    croiss(bio,Bi),
                    appartient(Biologique_Bi,Bi),
                    appartient(Biologique_Bi,Es)),
                    croiss(bleu,Bl),
                    appartient(Bleuissement_Bl,Bl),
                    appartient(Bleuissement_Bl,Es),
                    croiss(def,De),
                    appartient(Defaut_De,De),
                    appartient(Defaut_De,Es),
                    croiss(humi,Hu),
                    appartient(Humidite_Hu,Hu),
                    appartient(Humidite_Hu,Es),
                    croiss(aret,Ar),
                    appartient(Arete_Ar,Ar),
                    appartient(Arete_Ar,Es),
                    croiss(pouss,Po),
                    appartient(Poussiere_Po,Po),
                    appartient(Poussiere_Po,Es),
                    croiss(prop,Pr),
                    appartient(Proprete_Pr,Pr),
                    appartient(Proprete_Pr,Es),
                    croiss(graisse,Gr),
                    appartient(Graisse_Gr,Gr),
                    appartient(Graisse_Gr,Es).

post(sechage ,[
                    [nature, [bois]],
                    [carspec, [prepa]],
                    [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,Biologique_Bi  ,Bleuissement_Bl
,cloquage_non  ,Defaut_De  ,humidite_non  ,Arete_Ar  ,Proprete_Pr
,Poussiere_Po  ,Graisse_Gr]]]) :-
                    etat_de_depart(Ed),
                    valeur_item([etatsurf,Es],Ed,1),
                    appartient(humidite_oui,Es),
                    (appartient(biologique_fort,Es),
                    concat([Biologique_Bi],[biologique_non],[]);
                    not(appartient(biologique_fort,Es)),
                    croiss(bio,Bi),
                    appartient(Biologique_Bi,Bi),
                    appartient(Biologique_Bi,Es)),
                    croiss(bleu,Bl),
                    appartient(Bleuissement_Bl,Bl),
                    appartient(Bleuissement_Bl,Es),
                    croiss(def,De),
                    appartient(Defaut_De,De),
                    appartient(Defaut_De,Es),
                    croiss(aret,Ar),
                    appartient(Arete_Ar,Ar),
```

```
                    appartient(Arete_Ar,Es),
                    croiss(pouss,Po),
                    appartient(Poussiere_Po,Po),
                    appartient(Poussiere_Po,Es),
                    croiss(prop,Pr),
                    appartient(Proprete_Pr,Pr),
                    appartient(Proprete_Pr,Es),
                    croiss(graisse,Gr),
                    appartient(Graisse_Gr,Gr),
                    appartient(Graisse_Gr,Es).


post(poncage_arete,[
                    [nature, [bois]],
                    [carspec, [prepa]],
                    [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,Biologique_Bi ,Bleuissement_Bl
,cloquage_non  ,Defaut_De  ,humidite_non  ,arete_non   ,Proprete_Pr
,Poussiere_Po ,Graisse_Gr]]]) :-
                    etat_de_depart(Ed),
                    valeur_item([etatsurf,Es],Ed,1),
                    appartient(arete_oui,Es),
                    (appartient(biologique_fort,Es),
                    concat([Biologique_Bi],[biologique_non],[]);
                    not(appartient(biologique_fort,Es)),
                    croiss(bio,Bi),
                    appartient(Biologique_Bi,Bi),
                    appartient(Biologique_Bi,Es)),
                    croiss(bleu,Bl),
                    appartient(Bleuissement_Bl,Bl),
                    appartient(Bleuissement_Bl,Es),
                    croiss(def,De),
                    appartient(Defaut_De,De),
                    appartient(Defaut_De,Es),
                    croiss(aret,Ar),
                    appartient(Arete_Ar,Ar),
                    appartient(Arete_Ar,Es),
                    croiss(pouss,Po),
                    appartient(Poussiere_Po,Po),
                    appartient(Poussiere_Po,Es),
                    croiss(prop,Pr),
                    appartient(Proprete_Pr,Pr),
                    appartient(Proprete_Pr,Es),
                    croiss(graisse,Gr),
                    appartient(Graisse_Gr,Gr),
                    appartient(Graisse_Gr,Es).


post(degraissage ,[
                    [nature, [bois]],
                    [carspec, [prepa]],
                    [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,Biologique_Bi ,Bleuissement_Bl
,cloquage_non ,Defaut_De  ,humidite_non  ,arete_non   ,Proprete_Pr
,poussiere_non ,graisse_non]]]) :-
```

```
                    etat_de_depart(Ed),
                    valeur_item([etatsurf,Es],Ed,1),
                    appartient(graisse_oui, Es),
                    (appartient(biologique_fort,Es),
                    concat([Biologique_Bi],[biologique_non],[]);
                    not(appartient(biologique_fort,Es)),
                    croiss(bio,Bi),
                    appartient(Biologique_Bi,Bi),
                    appartient(Biologique_Bi,Es)),
                    croiss(prop,Pr),
                    appartient(Proprete_Pr,Pr),
                    appartient(Proprete_Pr,Es),
                    croiss(bleu,Bl),
                    appartient(Bleuissement_Bl,Bl),
                    appartient(Bleuissement_Bl,Es),
                    croiss(def,De),
                    appartient(Defaut_De,De),
                    appartient(Defaut_De,Es),
                    croiss(humi,Hu),
                    appartient(Humidite_Hu,Hu),
                    appartient(Humidite_Hu,Es).
post(depoussierage,[
                    [nature, [bois]],
                    [carspec, [prepa]],
                    [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,Biologique_Bi ,Bleuissement_Bl
,cloquage_non ,Defaut_De  ,humidite_non  ,arete_non   ,Proprete_Pr
,poussiere_non ,graisse_non]]]) :-
                    etat_de_depart(Ed),
                    valeur_item([etatsurf,Es],Ed,1),
                    appartient(poussiere_oui, Es),
                    not(appartient(graisse_oui,Es)),
                    (appartient(biologique_fort,Es),
                    concat([Biologique_Bi],[biologique_non],[]);
                    not(appartient(biologique_fort,Es)),
                    croiss(bio,Bi),
                    appartient(Biologique_Bi,Bi),
                    appartient(Biologique_Bi,Es)),
                    croiss(prop,Pr),
                    appartient(Proprete_Pr,Pr),
                    appartient(Proprete_Pr,Es),
                    croiss(bleu,Bl),
                    appartient(Bleuissement_Bl,Bl),
                    appartient(Bleuissement_Bl,Es),
                    croiss(def,De),
                    appartient(Defaut_De,De),
                    appartient(Defaut_De,Es).

 post(lavage_eau_de_javel ,[
                    [nature, [bois]],
                    [carspec, [prepa]],
                    [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,Biologique_Bi ,bleuissement_non
,cloquage_non ,Defaut_De ,humidite_non ,arete_non ,proprete_oui
,poussiere_non ,graisse_non]]]) :-
```

```
                    etat_de_depart(Ed),
                    valeur_item([etatsurf,Es],Ed,1),
                    appartient(bleuissement_oui, Es),
                    (appartient(biologique_fort,Es),
                    concat([Biologique_Bi],[biologique_non],[]);
                    not(appartient(biologique_fort,Es)),
                    croiss(bio,Bi),
                    appartient(Biologique_Bi,Bi),
                    appartient(Biologique_Bi,Es)),
                    croiss(def,De),
                    appartient(Defaut_De,De),
                    appartient(Defaut_De,Es).

post(lavage ,[
                    [nature, [bois]],
                    [carspec, [prepa]],
                    [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,Biologique_Bi ,bleuissement_non
,cloquage_non ,Defaut_De ,humidite_non ,arete_non ,proprete_oui
,poussiere_non ,graisse_non]]]) :-
                    etat_de_depart(Ed),
                    valeur_item([etatsurf,Es],Ed,1),
                    appartient(proprete_non, Es),
                    not(appartient(bleuissement_oui, Es)),
                    (appartient(biologique_fort,Es),
                    concat([Biologique_Bi],[biologique_non],[]);
                    not(appartient(biologique_fort,Es)),
                    croiss(bio,Bi),
                    appartient(Biologique_Bi,Bi),
                    appartient(Biologique_Bi,Es)),
                    croiss(def,De),
                    appartient(Defaut_De,De),
                    appartient(Defaut_De,Es).

post(xylamon_combi,[
                    [nature, [bois]],
                    [carspec, [prepa]],
                    [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,biologique_non ,bleuissement_non
,cloquage_non ,Defaut_De ,humidite_non ,arete_non ,proprete_oui
,poussiere_non ,graisse_non]]]) :-
                    etat_de_depart(Ed),
                    valeur_item([etatsurf,Es],Ed,1),
                    appartient(biologique_faible, Es),
                    croiss(def,De),
                    appartient(Defaut_De,De),
                    appartient(Defaut_De,Es).

post(lux_extra_blanc ,[
                    [nature, [bois]],
                    [carspec, [prepa]],
                    [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,biologique_non ,bleuissement_non
,cloquage_non ,defaut_non ,humidite_non ,arete_non ,proprete_oui
,poussiere_non ,graisse_non]],
```

```
                [aspect, [opaque]]]) :-
                etat_de_depart(Ed),
                valeur_item([etatsurf,Es],Ed,1),
                appartient(defaut_oui, Es).
post(porfilline ,[
                [nature, [bois]],
                [carspec, [prepa]],
                [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,biologique_non ,bleuissement_non
,cloquage_non ,defaut_non ,humidite_non ,arete_non ,proprete_oui
,poussiere_non ,graisse_non]],
                [aspect, [translucide]]]) :-
                etat_de_depart(Ed),
                valeur_item([etatsurf,Es],Ed,1),
                appartient(defaut_oui, Es).


post( silvanol ,[
                [nature, [bois]],
                [env, [ext]],
                [etatsurf, [recouvrable]],
                [aspect,  [translucide ,satine ,filmogene ,lisse
                          ,colore , filmogene]],
                [protect, [bleu ,pourri ,uv]],
                [resist, [eau]]]).

post( silva+gloss ,[
              [nature, [bois]],
              [env, [int]],
              [etatsurf, [recouvrable]],
              [carspec, [antiderapant]],
              [aspect, [translucide ,brillant ,lisse ,filmogene]],
              [resist, [abrasionforte ,lavage]]]).

post( silva+satin ,[
              [nature, [bois]],
              [env, [int]],
              [etatsurf, [recouvrable]],
              [carspec, [antiderapant]],
              [aspect, [translucide ,satine ,lisse ,filmogene]],
              [resist, [abrasionforte ,lavage]]]).

post( silvatanegloss ,[
              [nature, [bois]],
              [env, [ext]],
              [etatsurf, [recouvrable]],
              [aspect, [translucide ,brillant ,lisse ,filmogene]],
              [protect, [uv]],
              [resist, [abrasionmoyenne ,lavage]]]).

post( silvatanesatin ,[
               [nature, [bois]],
               [env, [int]],
               [etatsurf, [recouvrable]],
               [aspect, [translucide ,satine ,lisse ,filmogene]],
```

```
                [protect, [uv]],
                [resist, [abrasionmoyenne ,lavage]]]).

post( stellapaint, [
                 [nature, [bois]],
                 [env, [int]],
                 [etatsurf, [recouvrable]],
                 [aspect, [colore,opaque,brillant,lisse,filmogene]],
                 [resist, [abrasionlegere,lavagemoyenne]]]).

post( stelprimer, [
                [nature, [bois]],
                [env, [ext]],
                [etatsurf, [recouvrable]],
                [aspect, [opaque,filmogene]],
                [resist, [eau]],
                [carspec, [primer]]]).

post( tpcextrapale ,[
                  [nature, [bois]],
                  [env, [int]],
                  [etatsurf, [recouvrable]],
                  [aspect   ,[translucide   ,brillant   ,vernis
                             ,lisse ,filmogene]],
                  [resist ,[abrasionlegere ,lavagemoyen]]]).

post( tpcmat, [
              [nature, [bois]],
              [env, [int]],
              [etatsurf, [recouvrable]],
              [aspect, [translucide ,vernis ,mat ,lisse,
                        filmogene]],
              [resist, [abrasionlegere ,lavagemoyen]]]).

post( tpcsatin, [
               [nature, [bois]],
               [env, [int]],
               [etatsurf, [recouvrable]],
               [aspect,   [translucide  ,vernis  ,satine   ,lisse
                          ,filmogene]],
               [resist, [abrasionlegere ,lavagemoyen]]]).

post( tpcyacht ,[
                [nature, [bois]],
                [env, [ext]],
                [etatsurf, [recouvrable]],
                [aspect,  [translucide  ,brillant ,vernis  ,lisse
                          ,filmogene]],
                [resist, [abrasionlegere ,lavagemoyen]]]).

post( xylacolor, [
                [nature, [bois]],
                [env, [ext]],
                [etatsurf, [recouvrable]],
                [aspect, [colore ,translucide, satine, filmogene]],
```

```
                [protect, [bleu,uv]]]).

post( xyladecor, [
                [nature, [bois]],
                [env, [ext]],
                [etatsurf, [recouvrable]],
                [aspect, [translucide ,colore ,satine ,non-filmogene,li
                [protect, [bleu ,pourri ,uv]],
                [resist, [eau]]]).

post( xylamonbrown, [
                  [nature, [bois]],
                  [env, [ext]],
                  [etatsurf, [recouvrable]],
                  [aspect, Asf],
                  [resist, Resf],
                  [protect, Prof],
                  [carspec, Carf],
                  [applipart, [jardin]]]) :- etat_final(Ef),
                                             extas(Ef, Asf),
                                             extres(Ef, Resf),
                                             extprot(Ef, Prof),
                                             extcarspec(Ef, Carf).

post( xylamoncharpente, [
                      [nature, [bois]],
                      [env, [int]],
                      [etatsurf, [recouvrable]],
                      [aspect, [non-filmogene]],
                      [protect, [bleu,pourri,insectes]],
                      [applipart, [charpente]]]).

post( xylamonprimer, [
                    [nature, [bois]],
                    [env, [ext]],
                    [etatsurf, [recouvrable]],
                    [aspect, [non-filmogene]],
                    [protect, [bleu ,pourri ,insectes]]]).




post(auber3, [
              [nature, [metaux ,bois]],
              [env, [ext,moyenagressif,urbain]],
              [aspect, [colore,lisse,satine,filmogene,opaque]],
              [etatsurf, [recouvrable]],
              [carspec, [primer]],
              [protect, [bleu]],
              [resist, [abrasionlegere ,lavage]]]).

post(aubexol, [
               [nature,[metaux,bois]],
```

```
                [env, [int ,tresagressif,indust]],
                [etatsurf, [recouvrable]],
                [aspect, [colore,brillant,lisse,filmogene ,opaque]],
                [carspec, [antiderapant]],
                [resist, [abrasionforte,solvants ,lavage]],
                [protect, [corrosion]]]).

post(ergesol, [
                [nature, [metaux,bois]],
                [env, [moyenagressif,int]],
                [etatsurf, [recouvrable]],
                [aspect, [colore,satine,lisse,filmogene,opaque]],
                [carspec, [antiderapant]],
                [resist, [abrasionforte]]]).

post(incralac, [
                 [nature, [cuivre,zinc,plomb,alu]],
                 [env, [urbain,marin,moyenagressif]],
                 [etatsurf, [recouvrable]],
                 [aspect, [translucide,vernis,brillant,filmogene,lisse]],
                 [resist, [abrasionmoyenne]],
                 [protect, [corrosion]]]).

post(multiprimer, [
                   [nature, [metaux,bois]],
                   [env, [ext,urbain,moyenagressif]],
                   [carspec, [primer]],
                   [etatsurf, [recouvrable]],
                   [aspect, [opaque ,lisse, mat]]]).

post(permacryl, [
                  [nature, [metaux ,bois]],
                  [env, [ext,urbain,moyenagressif]],
                  [etatsurf, [recouvrable]],
                  [aspect, [colore,opaque,brillant,filmogene,lisse]],
                  [protect, [bleu]],
                  [resist, [lavagemoyen,stageau,abrasionlegere]]]).

post(permaline, [
                  [nature, [metaux,bois]],
                  [env,[ext,urbain,moyenagressif]],
                  [etatsurf, [recouvrable]],
                  [aspect, [colore,opaque,lisse,brillant,filmogene]],
                  [resist, [abrasionlegere,lavage]]]).

post(plomoferrinen, [
                     [nature, [acier,galva]],
                     [env, [indust,moyenagressif]],
                     [etatsurf, [recouvrable]],
                     [aspect, [opaque,lisse,mat,filmogene]],
                     [protect, [corrosion]]]).

post(plomoferrines, [
                     [nature, [acier,galva]],
                     [env, [indust,moyenagressif]],
```

```
                    [etatsurf, [recouvrable]],
                    [aspect, [opaque,lisse,mat,filmogene]]]).

post(primersurfacer, [
                     [nature, [metaux,bois]],
                     [env, [moyenagressif,ext,urbain]],
                     [carspec, [primer]],
                     [etatsurf, [recouvrable]],
                     [aspect, [lisse,mat,opaque,filmogene]],
                     [protect, [pourri]]]).

post(rubbercoating,[
                   [nature, [galva, acier, zinc]],
                   [env, [tresagressif, indust]],
                   [carspec, [primer]],
                   [etatsurf, [recouvrable]],
                   [aspect, [opaque,mat,lisse,filmogene]],
                   [protect, [corrosion]]]).

post(steldurplus,[
                  [nature, [metaux,bois]],
                  [env, [int,tresagressif,indust]],
                  [carspec, [antiderapant]],
                  [etatsurf, [recouvrable]],
                  [aspect, [colore,satine,opaque,lisse,filmogene]],
                  [resist, [lavage,abrasionforte,solvants]]]).

post(stellor, [
                  [nature, [metaux,bois]],
                  [env, [ext,tresagressif,urbain]],
                  [etatsurf, [recouvrable]],
                  [aspect, [brillant,opaque,lisse,filmogene]],
                  [protect, [uv]],
                  [resist, [solvants,lavage]]]).

post(steloxine, [
                   [nature, [acier,galva]],
                   [env, [moyenagressif,urbain]],
                   [etatsurf, [recouvrable]],
                   [aspect, [colore,opaque,satine,lisse,filmogene]],
                   [protect, [corrosion]],
                   [resist, [eau]]]).

post(trichrom, [
                   [nature, [metaux]],
                   [env, [tresagressif,marin,urbain]],
                   [etatsurf, [recouvrable]],
                   [aspect, [opaque,mat,lisse,filmogene]],
                   [protect, [corrosion]]]).

post(trisatin, [
                   [nature, [metaux,bois]],
                   [env, [ext,urbain,moyenagressif]],
                   [etatsurf, [recouvrable]],
                   [aspect, [colore,opaque,satine,lisse,filmogene]],
```

```
                [resist, [lavage,abrasionlegere]]]).

post(washprimer, [
                  [nature, [acier,galva]],
                  [env, [moyenagressif,urbain]],
                  [etatsurf, [recouvrable]],
                  [aspect, [satine,lisse,filmogene]],
                  [protect, [corrosion]]]).

post(zincano, [
                [nature, [acier,galva]],
                [env, [moyenagressif,urbain]],
                [etatsurf, [recouvrable]],
                [aspect, [mat,lisse,filmogene]],
                [protect, [corrosion]]]).

post(zincanoe, [
                 [nature, [acier,galva]],
                 [env, [moyenagressif,urbain]],
                 [etatsurf, [recouvrable]],
                 [aspect, [mat,lisse,filmogene]],
                 [protect, [corrosion]]]).




/*                PRECONDITIONS                                    */


pre( polissage, [
                  [etatsurf, Es]
                 ]):-
                   etat_de_depart( Ed),
                   extnat( Ed, [cuivre]),
                   valeur_item([etatsurf, Es], Ed, 1).

pre( degraissage, [
                    [etatsurf, [corrosion_oui,graisse_oui,
                                poussiere_non,proprete_oui,
                                accrochage_douteux,nettete_nu]]
                   ]):-
                   etat_de_depart(Ed),
                   extnat( Ed, [alu]),
                   valeur_item([etatsurf, Es], Ed, 1),
                   appartient( corrosion_oui, Es).

pre( degraissage, [
                    [etatsurf, [corrosion_non,graisse_oui,
                                poussiere_non,proprete_oui,
                                accrochage_douteux,nettete_nu]]
                   ]) :-
                   etat_de_depart(Ed),
                   extnat( Ed, [alu]).
```

```
pre( degraissage, [
                    [etatsurf, [corrosion_non, graisse_oui,
                                poussiere_non, proprete_oui,
                                accrochage_oui, nettete_nu]]
                   ]):-
                   etat_de_depart( Ed),
                   extnat( Ed, [acier]),
                   valeur_item([etatsurf, Es], Ed, 1),
                   appartient(nettete_nu, Es).

pre( degraissage, [
                    [etatsurf, [corrosion_non, graisse_oui,
                                poussiere_non, proprete_oui,
                                accrochage_oui, nettete_bon_etat]]
                   ]):-
                   etat_de_depart( Ed),
                   extnat( Ed, [Nature]),
                   appartient( Nature, [acier, alu, cuivre, galva, zinc])
                   valeur_item([etatsurf, Es], Ed, 1),
                   appartient(nettete_bon_etat, Es).

pre( degraissage, [
                    [etatsurf, [corrosion_non, graisse_oui,
                                poussiere_non, proprete_oui,
                                accrochage_oui, nettete_nu]]
                   ]):-
                   etat_de_depart(Ed),
                   extnat( Ed, [Nature]),
                   appartient( Nature, [galva, zinc]),
                   valeur_item([etatsurf, Es], Ed, 1),
                   appartient(nettete_nu, Es).


pre( lavage_eau_claire, [
                          [etatsurf, [corrosion_non, graisse_oui,
                                      Poussiere_A, proprete_non,
                                      Accrochage_B, nettete_bon_etat]]
                         ]):-
                   etat_de_depart( Ed),
                   extnat( Ed, [Nature]),
                   membre( Nature, [metaux]),
                   valeur_item( [etatsurf, Es], Ed, 1),
                   appartient(nettete_bon_etat, Es),
                   croiss(accro, B),
                   appartient(Accrochage_B, B),
                   appartient(Accrochage_B, Es),
                   croiss(pouss, A),
                   appartient(Poussiere_A, A),
                   appartient(Poussiere_A, Es).

pre( lavage_eau_claire, [
                          [etatsurf, [corrosion_non, graisse_oui,
                                      poussiere_oui, proprete_non,
                                      Accrochage_A, nettete_bon_etat]]
```

```
                        ]):-
                    etat_de_depart( Ed),
                    extnat( Ed, [Nature]),
                    membre( Nature, [metaux]),
                    valeur_item([etatsurf, Es], Ed, 1),
                    appartient(nettete_bon_etat, Es),
                    croiss(accro, A),
                    appartient(Accrochage_A,A),
                    appartient(Accrochage_A, Es).

pre( lavage_eau_claire, [
                         [etatsurf, [corrosion_non, graisse_oui,
                                     Poussiere_A, proprete_non,
                                     accrochage_oui, nettete_nu]]
                        ]):-
                    etat_de_depart( Ed),
                    extnat( Ed, [Nature]),
                    appartient( Nature, [acier, galva, zinc]),
                    appartient(accrochage_oui, Es),
                    croiss(pouss, A),
                    appartient(Poussiere_A, A),
                    appartient(Poussiere_A, Es).

pre( lavage_eau_claire, [
                         [etatsurf, [corrosion_non, graisse_oui,
                                     poussiere_non, proprete_non,
                                     accrochage_oui, nettete_nu]]
                        ]):-
                    etat_de_depart( Ed),
                    extnat( Ed, [Nature]),
                    appartient( Nature, [galva, zinc]),
                    valeur_item( [etatsurf, Es], Ed, 1),
                    inclus([accrochage_non, nettete_nu], Es).


pre( mordancage, [
                    [etatsurf, [corrosion_non,Graisse_A,
                                Poussiere_B, Proprete_C,
                                accrochage_non, nettete_nu]]
                   ]):-
                    etat_de_depart( Ed),
                    extnat( Ed, [Nature]),
                    appartient( Nature, [galva, zinc]),
                    valeur_item([etatsurf, Es], Ed, 1),
                    inclus([nettete_nu, corrosion_non], Es),
                    croiss(graisse, A),
                    appartient(Graisse_A, A),
                    appartient(Graisse_A, Es),
                    croiss(pouss, B),
                    appartient(Poussiere_B, B),
                    appartient(Poussiere_B, Es),
                    croiss(prop, C),
                    appartient(Proprete_C, C),
                    appartient(Proprete_C,Es).
```

```
pre( lavage_eau_savon, [
                        [etatsurf, [corrosion_non, Graisse_A,
                                    Poussiere_B, Proprete_C,
                                    accrochage_oui, nettete_nu]]
                    ]):-
                  etat_de_depart( Ed),
                  extnat( Ed, [zinc]),
                  valeur_item([etatsurf, Es], Ed, 1),
                  appartient(nettete_nu, Es),
                  croiss(graisse, A),
                  appartient(Graisse_A, A),
                  appartient(Graisse_A, Es),
                  croiss(pouss, B),
                  appartient(Poussiere_B, B),
                  appartient(Poussiere_B, Es),
                  croiss(prop, C),
                  appartient(Proprete_C, C),
                  appartient(Proprete_C,Es).


pre( epoussetage, [
                   [etatsurf, [corrosion_non, graisse_oui,
                               poussiere_oui, proprete_oui,
                               accrochage_oui, nettete_bon_etat]]
                  ]):-
                  etat_de_depart(Ed),
                  extnat( Ed, [Nature]),
                  membre( Nature, [metaux]),
                  valeur_item([etatsurf, Es], Ed, 1),
                  appartient(nettete_bon_etat, Es).

pre( epoussetage, [
                   [etatsurf, [corrosion_non, graisse_oui,
                               poussiere_oui, proprete_oui,
                               accrochage_oui, nettete_nu]]
                  ]) :-
                  etat_de_depart( Ed),
                  extnat( Ed, [acier]).

pre( epoussetage, [
                   [etatsurf, [corrosion_non, graisse_oui,
                               poussiere_oui, proprete_oui,
                               accrochage_non, nettete_nu]]
                  ]):-
                  etat_de_depart( Ed),
                  extnat( Ed, [Nature]),
                  appartient( Nature, [galva, zinc]),
                  valeur_item([etat_surf, Es], Ed, 1),
                  appartient(accrochage_non, Es).

pre( epoussetage, [
                   [etatsurf, [corrosion_oui,graisse_oui,
                               poussiere_oui,proprete_oui,
                               accrochage_douteux,nettete_nu]]
```

```
                    ]):-
                    etat_de_depart(Ed),
                    extnat( Ed, [alu]),
                    valeur_item([etatsurf, Es], Ed, 1),
                    appartient(corrosion_oui, Es).

pre( epoussetage, [
                     [etatsurf, [corrosion_non,graisse_oui,
                                 poussiere_oui,proprete_oui,
                                 accrochage_douteux, nettete_nu]]
                    ]) :-
                    etat_de_depart(Ed),
                    extnat( Ed, [alu]).

pre( epoussetage, [
                     [etatsurf, [corrosion_non, graisse_oui,
                                 poussiere_oui, proprete_oui,
                                 accrochage_oui, nettete_nu]]
                    ]):-
                    etat_de_depart(Ed),
                    extnat( Ed, [Nature]),
                    appartient( Nature, [galva, zinc]).


pre( derouillage, [
                     [etatsurf, [corrosion_oui, graisse_oui,
                                 Poussiere_B, Proprete_C,
                                 accrochage_oui, nettete_nu]]
                    ]):-
                    etat_de_depart(Ed),
                    extnat( Ed, [Nature]),
                    appartient( Nature, [galva,acier]),
                    valeur_item([etatsurf, [corrosion_oui, graisse_oui,
                                            Poussiere_B, Proprete_C,
                                            accrochage_oui, nettete_nu]],

pre( poncage, [
                    [etatsurf, [corrosion_non, graisse_oui,
                                poussiere_non, proprete_oui,
                                accrochage_non, nettete_bon_etat]]
                   ]):-
                       etat_de_depart( Ed),
                       extnat( Ed, [Nature]),
                       membre( Nature, [metaux]),
                       valeur_item([etatsurf, Es], Ed, 1),
                       appartient(proprete_non, Es).

pre( poncage, [
                    [etatsurf, [corrosion_non, graisse_oui,
                                Poussiere_A, proprete_oui,
                                accrochage_non, nettete_bon_etat]]
                   ]):-
                       etat_de_depart( Ed),
                       extnat( Ed, [Nature]),
                       membre( Nature, [metaux]),
```

```
                    valeur_item([etatsurf, Es], Ed, 1),
                    appartient(proprete_oui, Es),
                    croiss(pouss, A),
                    appartient(Poussiere_A, A),
                    appartient(Poussiere_A, Es).


pre( appel_specialiste ,[
                    [etatsurf , Es ]]) :-
                   etat_de_depart(Ed),
                   extnat( Ed, [bois]),
                   valeur_item([etatsurf,Es],Ed,1).

pre(decapage ,[
                   [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,Biologique_Bi ,Bleuissement_Bl
,cloquage_fort  ,Defaut_De  ,Humidite_Hu  ,Arete_Ar   ,Proprete_Pr
,Poussiere_Po ,Graisse_Gr]]]) :-
                   etat_de_depart(Ed),
                   extnat( Ed, [bois]),
                   valeur_item([etatsurf,Es],Ed,1),
                   (appartient(biologique_fort,Es),
                   concat([Biologique_Bi],[biologique_non],[]);
                   not(appartient(biologique_fort,Es)),
                   croiss(bio,Bi),
                   appartient(Biologique_Bi,Bi),
                   appartient(Biologique_Bi,Es)),
                   croiss(bleu,Bl),
                   appartient(Bleuissement_Bl,Bl),
                   appartient(Bleuissement_Bl,Es),
                   croiss(def,De),
                   appartient(Defaut_De,De),
                   appartient(Defaut_De,Es),
                   croiss(humi,Hu),
                   appartient(Humidite_Hu,Hu),
                   appartient(Humidite_Hu,Es),
                   croiss(aret,Ar),
                   appartient(Arete_Ar,Ar),
                   appartient(Arete_Ar,Es),
                   croiss(pouss,Po),
                   appartient(Poussiere_Po,Po),
                   appartient(Poussiere_Po,Es),
                   croiss(prop,Pr),
                   appartient(Proprete_Pr,Pr),
                   appartient(Proprete_Pr,Es),
                   croiss(graisse,Gr),
                   appartient(Graisse_Gr,Gr),
                   appartient(Graisse_Gr,Es).
pre(grattage ,[
                   [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,Biologique_Bi ,Bleuissement_Bl
,cloquage_faible  ,Defaut_De  ,Humidite_Hu  ,Arete_Ar   ,Proprete_Pr
,Poussiere_Po ,Graisse_Gr]]]) :-
                   etat_de_depart(Ed),
```

```
                    extnat( Ed, [bois]),
                    valeur_item([etatsurf,Es],Ed,1),
                    (appartient(biologique_fort,Es),
                    concat([Biologique_Bi],[biologique_non],[]);
                    not(appartient(biologique_fort,Es)),
                    croiss(bio,Bi),
                    appartient(Biologique_Bi,Bi),
                    appartient(Biologique_Bi,Es)),
                    croiss(bleu,Bl),
                    appartient(Bleuissement_Bl,Bl),
                    appartient(Bleuissement_Bl,Es),
                    croiss(def,De),
                    appartient(Defaut_De,De),
                    appartient(Defaut_De,Es),
                    croiss(humi,Hu),
                    appartient(Humidite_Hu,Hu),
                    appartient(Humidite_Hu,Es),
                    croiss(aret,Ar),
                    appartient(Arete_Ar,Ar),
                    appartient(Arete_Ar,Es),
                    croiss(pouss,Po),
                    appartient(Poussiere_Po,Po),
                    appartient(Poussiere_Po,Es),
                    croiss(prop,Pr),
                    appartient(Proprete_Pr,Pr),
                    appartient(Proprete_Pr,Es),
                    croiss(graisse,Gr),
                    appartient(Graisse_Gr,Gr),
                    appartient(Graisse_Gr,Es).
pre(sechage ,[
                    [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,Biologique_Bi  ,Bleuissement_Bl
,cloquage_non  ,Defaut_De  ,humidite_oui  ,Arete_Ar  ,Proprete_Pr
,Poussiere_Po  ,Graisse_Gr]]]) :-
                    etat_de_depart(Ed),
                    extnat( Ed, [bois]),
                    valeur_item([etatsurf,Es],Ed,1),
                    (appartient(biologique_fort,Es),
                    concat([Biologique_Bi],[biologique_non],[]);
                    not(appartient(biologique_fort,Es)),
                    croiss(bio,Bi),
                    appartient(Biologique_Bi,Bi),
                    appartient(Biologique_Bi,Es)),
                    croiss(bleu,Bl),
                    appartient(Bleuissement_Bl,Bl),
                    appartient(Bleuissement_Bl,Es),
                    croiss(def,De),
                    appartient(Defaut_De,De),
                    appartient(Defaut_De,Es),
                    croiss(aret,Ar),
                    appartient(Arete_Ar,Ar),
                    appartient(Arete_Ar,Es),
                    croiss(pouss,Po),
                    appartient(Poussiere_Po,Po),
                    appartient(Poussiere_Po,Es),
```

```
                    croiss(prop,Pr),
                    appartient(Proprete_Pr,Pr),
                    appartient(Proprete_Pr,Es),
                    croiss(graisse,Gr),
                    appartient(Graisse_Gr,Gr),
                    appartient(Graisse_Gr,Es).
pre(poncage_arete ,[
                    [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,Biologique_Bi ,Bleuissement_Bl
,cloquage_non  ,Defaut_De  ,humidite_non ,arete_oui   ,Proprete_Pr
,Poussiere_Po ,Graisse_Gr]]]) :-
                    etat_de_depart(Ed),
                    extnat( Ed, [bois]),
                    valeur_item([etatsurf,Es],Ed,1),
                    (appartient(biologique_fort,Es),
                    concat([Biologique_Bi],[biologique_non],[]);
                    not(appartient(biologique_fort,Es)),
                    croiss(bio,Bi),
                    appartient(Biologique_Bi,Bi),
                    appartient(Biologique_Bi,Es)),
                    croiss(bleu,Bl),
                    appartient(Bleuissement_Bl,Bl),
                    appartient(Bleuissement_Bl,Es),
                    croiss(def,De),
                    appartient(Defaut_De,De),
                    appartient(Defaut_De,Es),
                    croiss(pouss,Po),
                    appartient(Poussiere_Po,Po),
                    appartient(Poussiere_Po,Es),
                    croiss(prop,Pr),
                    appartient(Proprete_Pr,Pr),
                    appartient(Proprete_Pr,Es),
                    croiss(graisse,Gr),
                    appartient(Graisse_Gr,Gr),
                    appartient(Graisse_Gr,Es).
pre(depoussierage ,[
                    [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,Biologique_Bi ,Bleuissement_Bl
,cloquage_non  ,Defaut_De  ,humidite_non ,arete_non   ,Proprete_Pr
,poussiere_oui ,graisse_non]]]) :-
                    etat_de_depart(Ed),
                    extnat( Ed, [bois]),
                    valeur_item([etatsurf,Es],Ed,1),
                    (appartient(biologique_fort,Es),
                    concat([Biologique_Bi],[biologique_non],[]);
                    not(appartient(biologique_fort,Es)),
                    croiss(bio,Bi),
                    appartient(Biologique_Bi,Bi),
                    appartient(Biologique_Bi,Es)),
                    croiss(bleu,Bl),
                    appartient(Bleuissement_Bl,Bl),
                    appartient(Bleuissement_Bl,Es),
                    croiss(def,De),
                    appartient(Defaut_De,De),
                    appartient(Defaut_De,Es),
```

```
                    croiss(prop,Pr),
                    appartient(Proprete_Pr,Pr),
                    appartient(Proprete_Pr,Es).
pre(degraissage ,[
                    [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,Biologique_Bi ,Bleuissement_Bl
,cloquage_non  ,Defaut_De  ,humidite_non ,arete_non   ,Proprete_Pr
,Poussiere_Po ,graisse_oui]]]) :-
                    etat_de_depart(Ed),
                    extnat( Ed, [bois]),
                    valeur_item([etatsurf,Es],Ed,1),
                    (appartient(biologique_fort,Es),
                    concat([Biologique_Bi],[biologique_non],[]);
                    not(appartient(biologique_fort,Es)),
                    croiss(bio,Bi),
                    appartient(Biologique_Bi,Bi),
                    appartient(Biologique_Bi,Es)),
                    croiss(bleu,Bl),
                    appartient(Bleuissement_Bl,Bl),
                    appartient(Bleuissement_Bl,Es),
                    croiss(def,De),
                    appartient(Defaut_De,De),
                    appartient(Defaut_De,Es),
                    croiss(pouss,Po),
                    appartient(Poussiere_Po,Po),
                    appartient(Poussiere_Po,Es),
                    croiss(prop,Pr),
                    appartient(Proprete_Pr,Pr),
                    appartient(Proprete_Pr,Es).

pre(lavage_eau_de_javel ,[
                    [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,Biologique_Bi ,bleuissement_oui
,cloquage_non  ,Defaut_De  ,humidite_non ,arete_non   ,Proprete_Pr
,poussiere_non ,graisse_non]]]) :-
                    etat_de_depart(Ed),
                    extnat( Ed, [bois]),
                    valeur_item([etatsurf,Es],Ed,1),
                    (appartient(biologique_fort,Es),
                    concat([Biologique_Bi],[biologique_non],[]);
                    not(appartient(biologique_fort,Es)),
                    croiss(bio,Bi),
                    appartient(Biologique_Bi,Bi),
                    appartient(Biologique_Bi,Es)),
                    croiss(def,De),
                    appartient(Defaut_De,De),
                    appartient(Defaut_De,Es),
                    croiss(prop,Pr),
                    appartient(Proprete_Pr,Pr),
                    appartient(Proprete_Pr,Es).

pre(lavage ,[
                    [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,Biologique_Bi ,bleuissement_non
,cloquage_non  ,Defaut_De ,humidite_non ,arete_non ,proprete_non
```

```
,poussiere_non ,graisse_non]]]) :-
                etat_de_depart(Ed),
                extnat( Ed, [bois]),
                valeur_item([etatsurf,Es],Ed,1),
                (appartient(biologique_fort,Es),
                concat([Biologique_Bi],[biologique_non],[]);
                not(appartient(biologique_fort,Es)),
                croiss(bio,Bi),
                appartient(Biologique_Bi,Bi),
                appartient(Biologique_Bi,Es)),
                croiss(def,De),
                appartient(Defaut_De,De),
                appartient(Defaut_De,Es).

pre(xylamon_combi ,[
                [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,biologique_faible ,bleuissement_non
,cloquage_non  ,Defaut_De ,humidite_non ,arete_non ,proprete_oui
,poussiere_non ,graisse_non]]]) :-
                etat_de_depart(Ed),
                extnat( Ed, [bois]),
                valeur_item([etatsurf,Es],Ed,1),
                croiss(def,De),
                appartient(Defaut_De,De),
                appartient(Defaut_De,Es).

pre(lux_extra_blanc ,[
               [derntrt, [lux_extra_blanc]],
               [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,biologique_non ,bleuissement_non
,cloquage_non  ,defaut_oui ,humidite_non ,arete_non ,proprete_oui
,poussiere_non ,graisse_non]]]) :-
                etat_de_depart(Ed),
                extnat( Ed, [bois]).

pre(porfilline ,[
                [derntrt, [porfilline]],
                [etatsurf,[noeud_non, excudation_non,
porosite_non,  fissuration_non  ,biologique_non ,bleuissement_non
,cloquage_non  ,defaut_oui ,humidite_non ,arete_non ,proprete_oui
,poussiere_non ,graisse_non]]]) :-
                etat_de_depart(Ed),
                extnat( Ed, [bois]).



pre(silvanol, [
                [etatsurf, [recouvrable]],
                [derntrt, [silvanol]]]).


pre(silva+gloss, [
                [etatsurf, [recouvrable]],
                [derntrt, [silva+gloss]]]).
```

```
pre(silva+satin, [
                  [etatsurf, [recouvrable]],
                  [derntrt, [silva+satin]]]).

pre(silvatanegloss, [
                   [etatsurf, [recouvrable]],
                   [derntrt, [silvatanegloss]]]).

pre(silvatanesatin, [
                   [etatsurf, [recouvrable]],
                   [derntrt, [silvatanesatin]]]).

pre(stellapaint, [
                  [carspec, [primer]],
                  [etatsurf, [recouvrable]],
                  [derntrt, [stellapaint]]]).

pre(stelprimer, [
               [etatsurf, [recouvrable]],
               [derntrt, [stelprimer]]]).

pre(tpcextrapale, [
                  [etatsurf, [recouvrable]],
                  [derntrt, [tpcextrapale]]]).

pre(tpcmat, [
           [etatsurf, [recouvrable]],
           [derntrt, [tpcmat]]]).

pre(tpcsatin, [
             [etatsurf, [recouvrable]],
             [derntrt, [tpcsatin]]]).

pre(tpcyacht, [
                  [etatsurf, [recouvrable]],
                  [derntrt, [tpcyacht]]]).

pre(xylacolor, [
                  [etatsurf, [recouvrable]],
                  [derntrt, [xylacolor]]]).

pre(xyladecor, [
                  [etatsurf, [recouvrable]],
                  [derntrt, [xyladecor]]]).

pre(xylamonbrown, [
                 [etatsurf, [recouvrable]],
                 [derntrt, [xylamonbrown]]]).

pre(xylamoncharpente, [
                   [etatsurf, [recouvrable]],
                   [derntrt, [xylamoncharpente]]]).

pre(xylamonprimer, [
                  [etatsurf, [recouvrable]],
```

```
                [derntrt, [xylamonprimer]]]).


pre(auber3, [
            [etatsurf, [recouvrable]],
            [derntrt, [auber3]]]).

pre(aubexol, [
             [etatsurf, [recouvrable]],
             [derntrt, [aubexol]]]).

pre(ergesol, [
             [etatsurf, [recouvrable]],
             [derntrt, [ergesol]]]).

pre(incralac, [
              [etatsurf, [recouvrable]],
              [derntrt, [incralac]]]).

pre(multiprimer, [
                 [etatsurf, [recouvrable]],
                 [derntrt, [multiprimer]]]).

pre(permacryl, [
               [carspec, [primer]],
               [etatsurf, [recouvrable]],
               [derntrt, [permacryl]]]).

pre(permaline, [
               [carspec, [primer]],
               [etatsurf, [recouvrable]],
               [derntrt, [permaline]]]).

pre(plomoferrinen, [
                   [etatsurf, [recouvrable]],
                   [derntrt, [plomoferrinen]]]).

pre(plomoferrines, [
                   [etatsurf, [recouvrable]],
                   [derntrt, [plomoferrines]]]).

pre(primersurfacer, [
                    [etatsurf, [recouvrable]],
                    [derntrt, [primersurfacer]]]).

pre(rubbercoating, [
                   [etatsurf, [recouvrable]],
                   [derntrt, [rubbercoating]]]).

pre(steldurplus, [
                 [etatsurf, [recouvrable]],
                 [derntrt, [steldurplus]]]).

pre(stellor, [
```

```
                [etatsurf, [recouvrable]],
                [derntrt, [stellor]]]).

pre(steloxine, [
                 [etatsurf, [recouvrable]],
                 [derntrt, [steloxine]]]).

pre(trichrom, [
                [etatsurf, [recouvrable]],
                [derntrt, [trichrom]]]).

pre(trisatin, [
                [etatsurf, [recouvrable]],
                     [derntrt, [trisatin]]]).

pre(washprimer, [
                  [etatsurf, [recouvrable]],
                  [derntrt, [washprimer]]]).

pre(zincano, [
               [etatsurf, [recouvrable]],
               [derntrt, [zincano]]]).

pre(zincanoe, [
                [etatsurf, [recouvrable]],
                [derntrt, [zincanoe]]]).


/*                    COMPOSITIONS                                    */


comp(silvanol, [[s,hydrocarbali],[p,[silice]],[l,alkyde],
               [pvc,0.088]]).

comp(silva+gloss, [[s,hydrocarbarometesthers],[p,[]],[l,purmono],
                   [pvc,0.07]]).

comp(silva+satin, [[s,hydrocarbarometesthers],[p,[]],[l,purmono],
                  [pvc,0.07]]).

comp(silvatanegloss, [[s,hydrocarbali],[p,[]],[l,alkyde],
                     [pvc,0.072]]).

comp(silvatanesatin, [[s,hydrocarbali],[p,[silice]],[l,alkyde],
                     [pvc,0.072]]).

comp(stellapaint, [[s,hydrocarbali],[p,[]],[l,alkyde],
                  [pvc,0.14]]).

comp(stelprimer, [[s,hydrocarbali],[p,[dioxydetitane]],[l,alkyde],
                 [pvc,0.461]]).

comp(tpcextapale, [[s,hydrocarbali],[p,[]],[l,alkyde],
                  [pvc,0.05]]).
```

```
comp(tpcmat, [[s,hydrocarbali],[p,[]],[l,alkyde],
              [pvc,0.08]]).

comp(tpcsatin, [[s,hydrocarbali],[p,[]],[l,alkyde],
              [pvc,0.05]]).

comp(tpcyacht, [[s,hydrocarbali],[p,[]],[l,alkyde],
              [pvc,0.07]]).

comp(xylacolor, [[s,eau],[p,[oxydetitane,oxidefer]],[l,acrylique],
               [pvc,0.25]]).

comp(xyladecor, [[s,hydrocarbali],[p,[oxydetitane,oxydefer]],[l,alkyde],
               [pvc,0.088]]).

comp(xylamonbrown, [[s,hydrocarbali],[p,[dioxydetitane]],[l,alkyde],
                  [pvc,0.38]]).

comp(xylamoncharpente, [[s,hydrocarbali],[p,[dioxydetitane]],[l,alkyde],
                      [pvc,0.38]]).

comp(xylamonprimer, [[s,hydrocarbali],[p,[dioxydetitane]],[l,alkyde],
                   [pvc,0.5]]).



comp(auber3, [[s,hydrocarbali],[p, [dioxydetitane, silicesdediatomee]],
            [l,alkyde],[pvc,0.305]]).

comp(aubexol, [[s,hydrocarbarometesthers],[p,[]],[l,purmono],
              [pvc,0.169]]).

comp(ergesol, [[s,hydrocarbali],[p,dioxydetitane],[l,alkyde],
              [pvc,0.261]]).

comp(incralac, [[s,hydrocarbarometesthers],[p,[]],[l,acrylique],
              [pvc,0.000]]).

comp(multiprimer, [[s,eau],[p,[dioxydetitane]],[l,acrylique],
                 [pvc,0.289]]).

comp(permacryl, [[s,eau],[p,[dioxydetitane]],[l,acrylique],
               [pvc,0.206]]).

comp(permaline, [[s,hydrocarbali],[p,[]],[l,alkyde],
               [pvc,0.153]]).

comp(plomoferrinen, [[s,hydrocarbali],[p,[miniumplomb,oxydefer]],
                   [l,alkyde],[pvc,1.000]]).

comp(plomoferrines, [[s,hydrocarbali],[p,[miniumplomb,oxydefer]],
                   [l,alkyde],[pvc,1.000]]).

comp(primersurfacer, [[s,hydrocarbali],[p,[dioxydetitane,silicesdediatom
```

```
                    [l,alkyde],[pvc,0.355]]).

comp(rubbercoating, [[s,chlores],[p,[]],[l,purmono],[pvc,1.000]]).

comp(steldurplus, [[s,hydrocarbarometesthers],[p,[]],[l,purmono],
              [pvc,0.145]]).

comp(stellor, [[s,hydrocarbali],[p,[oxydetitane]],[l,alkyde],
             [pvc,0.174]]).

comp(steloxine, [[s,hydrocarbali],[p,[phosphatezinc]],[l,alkyde],
              [pvc,0.401]]).

comp(trichrom, [[s,hydrocarbali],[p,[chromatezinc,dioxydetitane,oxydefer
              [l,alkyde],[pvc,0.518]]).

comp(trisatin, [[s,hydrocarbali],[p,[oxydetitane,carbonatecalcium]],
              [l,alkyde],[pvc,0.385]]).

comp(washprimer, [[s,hydrocarbarometalcools],[p,[chromatezinc]],
               [l,epoxyetphenolique],[pvc,1.000]]).

comp(zincano, [[s,hydrocarbarom],[p,[zinc]],[l,vinylchlore],
             [pvc,0.715]]).

comp(zincanoe, [[s,hydrocarbarom],[p,[zinc]],[l,epoxypolyamide],
              [pvc,1.000]]).

/*                    EFFET DE BORD                                    */


effetdebord(auber3  ,  [[pm,51],  [tsech,[4,15]],  [comp,mono],
[base,s],  [prom,0],  [fac,oui], [stock,1], [odeur,o], [anci,a],
[maplir,o], [maplip,n],[qual,s]]).

effetdebord(aubexol  ,  [[pm,129],  [tsech,[15,15]],  [comp,mono],
[base,s],  [prom,0],  [fac,oui], [stock,1], [odeur,o], [anci,a],
[maplir,o], [maplip,o],[qual,s]]).

effetdebord(ergesol  ,  [[pm,62],  [tsech,[15,15]],  [comp,mono],
[base,s],  [prom,0],  [fac,oui], [stock,1], [odeur,o], [anci,a],
[maplir,o], [maplip,n],[qual,c]]).

effetdebord(incralac  ,  [[pm,26],  [tsech,[1,1]],  [comp,mono],
[base,s],  [prom,0],  [fac,oui], [stock,1], [odeur,o], [anci,a],
[maplir,o], [maplip,n],[qual,s]]).

effetdebord(lux_extra_blanc  ,  [[pm,185],  [tsech,[0,0]],
[comp,mono],  [base,s],  [prom,0],  [fac,oui],  [stock,1],
[odeur,o], [anci,a], [maplir,n], [maplip,n],[qual,s]]).

effetdebord(multiprimer  ,  [[pm,48],  [tsech,[2,5]],  [comp,mono],
[base,a],  [prom,0],  [fac,d],  [stock,1],  [odeur,n],  [anci,n],
[maplir,o], [maplip,n],[qual,s]]).
```

```
effetdebord(permacryl , [[pm,35], [tsech,[6,6]], [comp,mono],
[base,a], [prom,0], [fac,d], [stock,1], [odeur,n], [anci,n],
[maplir,o], [maplip,n],[qual,s]]).

effetdebord(permaline , [[pm,39], [tsech,[20,20]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,o], [anci,a],
[maplir,o], [maplip,o],[qual,s]]).

effetdebord(plomoferinen , [[pm,28], [tsech,[15,15]],
[comp,mono], [base,s], [prom,0], [fac,oui], [stock,1],
[odeur,o], [anci,a], [maplir,n], [maplip,n],[qual,c]]).

effetdebord(plomoferines , [[pm,33], [tsech,[15,15]],
[comp,mono], [base,s], [prom,0], [fac,oui], [stock,1],
[odeur,o], [anci,a], [maplir,n], [maplip,n],[qual,s]]).

effetdebord(porfilline , [[pm,351], [tsech,[0,0]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,o], [anci,a],
[maplir,n], [maplip,n], [qual,s]]).

effetdebord(primersurfacer , [[pm,40], [tsech,[15,15]],
[comp,mono], [base,s], [prom,0], [fac,oui], [stock,1],
[odeur,o], [anci,a], [maplir,o], [maplip,n], [qual,s]]).

effetdebord(sivanol , [[pm,49], [tsech,[15,15]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,o], [anci,n],
[maplir,n], [maplip,n], [qual,s]]).

effetdebord(silva+gloss , [[pm,56], [tsech,[4,4]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,o], [anci,n],
[maplir,o], [maplip,o], [qual,s]]).

effetdebord(silva+satin , [[pm,76], [tsech,[16,16]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,o], [anci,n],
[maplir,o], [maplip,o], [qual,s]]).

effetdebord(sivatanegloss , [[pm,41], [tsech,[8,8]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,o], [anci,n],
[maplir,o], [maplip,n], [qual,s]]).

effetdebord(silvatanesatin , [[pm,41], [tsech,[8,8]],
[comp,mono], [base,s], [prom,0], [fac,oui], [stock,1],
[odeur,o], [anci,n], [maplir,o], [maplip,n], [qual,s]]).

effetdebord(steldurplus , [[pm,97], [tsech,[5,48]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,o], [anci,n],
[maplir,o], [maplip,o], [qual,s]]).

effetdebord(stellapaint , [[pm,31], [tsech,[15,15]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,o], [anci,a],
[maplir,o], [maplip,o], [qual,c]]).

effetdebord(stellor , [[pm,56], [tsech,[4,4]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,o], [anci,a],
[maplir,o], [maplip,o], [qual,s]]).
```

```
effetdebord(stelloxine , [[pm,57], [tsech,[18,18]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,o], [anci,n],
[maplir,o], [maplip,n], [qual,s]]).

effetdebord(stelprimer , [[pm,45], [tsech,[15,15]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,o], [anci,a],
[maplir,o], [maplip,n], [qual,c]]).

effetdebord(tpcextrapale , [[pm,27], [tsech,[15,15]],
[comp,mono], [base,s], [prom,0], [fac,oui], [stock,1],
[odeur,o], [anci,a], [maplir,o], [maplip,o], [qual,c]]).

effetdebord(tpcmat , [[pm,39], [tsech,[15,15]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,o], [anci,a],
[maplir,o], [maplip,o], [qual,c]]).

effetdebord(tpcsatin , [[pm,39], [tsech,[15,15]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,o], [anci,a],
[maplir,o], [maplip,o], [qual,c]]).

effetdebord(tpcyacht , [[pm,27], [tsech,[15,15]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,o], [anci,a],
[maplir,o], [maplip,n], [qual,c]]).

effetdebord(trichrom , [[pm,36], [tsech,[18,18]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,o], [anci,a],
[maplir,n], [maplip,n], [qual,s]]).

effetdebord(trisatin , [[pm,44], [tsech,[15,15]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,o], [anci,a],
[maplir,o], [maplip,o], [qual,s]]).

effetdebord(washprimer , [[pm,41], [tsech,[1,1]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,non], [anci,a],
[maplir,o], [maplip,o], [qual,s]]).

effetdebord(xylacolor , [[pm,29], [tsech,[0,1]], [comp,mono],
[base,a], [prom,0], [fac,oui], [stock,1], [odeur,non], [anci,n],
[maplir,o], [maplip,n], [qual,s]]).

effetdebord(xyladecor , [[pm,27], [tsech,[18,24]], [comp,mono],
[base,s], [prom,0], [fac,oui], [stock,1], [odeur,o], [anci,a],
[maplir,o], [maplip,n], [qual,c]]).

effetdebord(xylamonbrown , [[pm,71], [tsech,[24,24]],
[comp,mono], [base,s], [prom,0], [fac,oui], [stock,1],
[odeur,o], [anci,a], [maplir,n], [maplip,n], [qual,s]]).

effetdebord(xylamoncharpente , [[pm,47], [tsech,[240,240]],
[comp,mono], [base,s], [prom,0], [fac,oui], [stock,1],
[odeur,o], [anci,a], [maplir,n], [maplip,n], [qual,s]]).

effetdebord(xylamoncombi , [[pm,118], [tsech,[192,192]],
[comp,mono], [base,s], [prom,0], [fac,oui], [stock,1],
```

```
[odeur,non], [anci,a], [maplir,n], [maplip,n], [qual,s]]).

effetdebord(xylamonprimer     ,     [[pm,58],     [tsech,[18,24]],
[comp,mono],    [base,s],    [prom,0],    [fac,oui],    [stock,1],
[odeur,non], [anci,a], [maplir,n], [maplip,n], [qual,s]]).

effetdebord(zincano   ,  [[pm,99],  [tsech,[14,14]],  [comp,mono],
[base,s], [prom,0],  [fac,oui], [stock,1], [odeur,o], [anci,a],
[maplir,o], [maplip,n], [qual,s]]).

/* valeur des différerntes classes    */

/*  DEUXIEME CLASSE DE FAITS                                        */


/*              CARACTERISTIQUES DES VALEURS DU VECTEUR D'ETAT     */

/*                    ORDRE DES VALEURS DE CRITERE POUR CHAQUE
                                        NOM DE CRITERE             */

croiss( forme, [horizontal,oblique,vertical]).
croiss( forme, [plan,tubulaire,angulaire]).
croiss( forme, [surfvershaut,surfversbas]).

croiss( env, [rural,urbain,indust]).
croiss( env, [peuagressif,moyenagressif,tresagressif]).
croiss( env, [int,ext]).

croiss( carspec, [salissant,peusalissant]).
croiss( carspec, [toxique,nontoxique]).
croiss( carspec, [derapant,antiderapant]).

croiss( etatsurf, [peurouille,voilerouille,rouilleimport,oxyde]).
croiss( pouss, [poussiere_oui, poussiere_non]).
croiss( prop, [proprete_oui, proprete_non]).
croiss( graisse, [graisse_oui, graisse_non]).
croiss( accro, [accrochage_oui,accrochage_douteux,accrochage_non]).
croiss(bleu,[bleuissement_oui,bleuissement_non]).
croiss(cloq,[cloquage_fort,cloquage_faible,cloquage_non]).
croiss(bio,[biologique_fort,biologique_faible,biologique_non]).
croiss(def,[defaut_oui,defaut_non]).
croiss(humi,[humidite_oui,humidite_non]).
croiss(aret,[arete_oui,arete_non]).

croiss( resist, [rural,urbain,indust]).
croiss( resist, [marin]).
croiss( resist, [stageau]).
croiss( resist, [peuagressif,moyenagressif,tresagressif]).
croiss( resist, [abrasionlegere, abrasionmoyenne, abrasionforte]).
croiss( resist, [eau, lavagemoyen, lavage]).
croiss( resist, [solvants]).

/* TROISIEME CLASSE DE FAITS                                        */
```

```
/*              CLASSES DE VALEURS DE CRITERE POUR CHAQUE
                                                NOM DE CRITERE          */

itemvalclasse( aspect, [translucide, incolore, lisse, non-filmogene], pe
itemvalclasse(aspect, [opaque ,mat, satine, brillant, special, peint, ve
itemvalclasse( aspect, [colore, structure, filmogene], supp).
itemvalclasse( resist, [vapeau], perm).
itemvalclasse( applipart, [jardin], perm).


/*                  CARACTERISTIQUES DES COMPOSANTS                     */

/*                    ECHELLE D'AGRESSIVITE DES SOLVANTS                */

echelagr([[eau,1], [hydrocarbali, 2], [alcools, 3], [hydrocarbarom, 4],
          [hydrocarbarometalcools, 4], [esthers, 5], [hydrocarbarometest
          [cetones, 6], [chlores, 7]]).


/*                    ECHELLE DE THERMOPLASTICITE DES LIANTS            */

echeltherm([[eau, 1], [caoutchlore, 2], [acrylique, 3], [alkyde, 4],
            [vinylchlore, 4], [purmono, 5], [epoxy, 6], [epoxyetphenoliq
            [epoxypolyamide, 6], [purbi, 7]]).

/*                    LIANTS SAPONIFIABLES                              */

saponifiable( [alkyde, epoxyetphenolique]).

/*                    PIGMENTS COMPOSES DE ZINC                         */

compzinc( [zinc, chromatezinc, phosphatezinc]).


/* QUATRIEME CLASSE DE FAITS                                            */

/* COEFFICIANT D'IMPORTANCE ET DE PONDERATION                           */

coeffimp([[pm,1],[comp,11],   [prom,7],   [tsech,8],   [fac,1],
[odeur,20], [anc,6], [maplir,2], [maplip,3] ,[qual,20], [npr,10]]).

coeffpond([[pm,1],[comp,20],  [prom,10], [tsech,5], [fac,20],
[odeur,20],  [anc,20],  [maplir,20], [maplip,20] ,[qual,20],
[npr,20]]).



/* QUESTIONNAIRE QUESTIONNAIRE QUESTIONNAIRE QUESTIONNAIRE */

question([[quest,['L''odeur',vous,derange,'t-elle?']],[val,o],
[car,odeur], [evalu,Ev]]) :-
       coeffimp([[pm,_],[comp,_],     [prom,_],    [tsech,_],
[fac,_],  [odeur,I],  [anc,_], [maplir,_], [maplip,_] ,[qual,_],
[npr,_]]),
       coeffpond([[pm,_],[comp,_],   [prom,_],   [tsech,_],
```

```
[fac,_], [odeur,P], [anc,_], [maplir,_], [maplip,_] ,[qual,_],
[npr,_]]),
        Ev = I * P.

question([[quest,['Pour',un,travail,'soigne?']],        [val,c],
[car,qual], [evalu,Ev]]) :-
      coeffimp([[pm,_],[comp,_],     [prom,_],    [tsech,_],
[fac,_], [odeur,_], [anc,_], [maplir,_], [maplip,_] ,[qual,I],
[npr,_]]),
      coeffpond([[pm,_],[comp,_],   [prom,_],   [tsech,_],
[fac,_], [odeur,_], [anc,_], [maplir,_], [maplip,_] ,[qual,P],
[npr,_]]),
        Ev = I * P.
question([[quest,['Pour',un,systeme,sans,nouveaux,'produits?']],
[val,n], [car,anc], [evalu,Ev]]) :-
      coeffimp([[pm,_],[comp,_],     [prom,_],    [tsech,_],
[fac,_], [odeur,_], [anc,I], [maplir,_], [maplip,_] ,[qual,_],
[npr,_]]),
      coeffpond([[pm,_],[comp,_],   [prom,_],   [tsech,_],
[fac,_], [odeur,_], [anc,P], [maplir,_], [maplip,_] ,[qual,_],
[npr,_]]),
        Ev = I * P.
question([[quest,['Desirez-vous',utilisez,un,'pistolet?']],
[val,n], [car,maplip], [evalu,Ev]]) :-
      coeffimp([[pm,_],[comp,_],     [prom,_],    [tsech,_],
[fac,_], [odeur,_], [anc,_], [maplir,_], [maplip,I] ,[qual,_],
[npr,_]]),
      coeffpond([[pm,_],[comp,_],   [prom,_],   [tsech,_],
[fac,_], [odeur,_], [anc,_], [maplir,_], [maplip,P] ,[qual,_],
[npr,_]]),
        Ev = I * P.
question([[quest,['Desirez-vous',utilisez,un,rouleau,'?']],
[val,n], [car,maplir], [evalu,Ev]]) :-
      coeffimp([[pm,_],[comp,_],     [prom,_],    [tsech,_],
[fac,_], [odeur,_], [anc,_], [maplir,I], [maplip,_] ,[qual,_],
[npr,_]]),
      coeffpond([[pm,_],[comp,_],   [prom,_],   [tsech,_],
[fac,_], [odeur,_], [anc,_], [maplir,P], [maplip,_] ,[qual,_],
[npr,_]]),
        Ev = I * P.


/* CLASSE                                                          */


classe(bois,    [[pm,27], [tsech,1], [prom,0], [comp,mono],
[fac,oui]]).

classe(metaux,   [[pm,35], [tsech,5], [prom,0], [comp,mono],
[fac,oui]]).

classe(acier,   [[pm,28], [tsech,5], [prom,0], [comp,mono],
[fac,oui]]).

classe(galva,   [[pm,28], [tsech,5], [prom,0], [comp,mono],
```

```
[fac,oui]]).

classe(cuivre,  [[pm,26], [tsech,5], [prom,0], [comp,mono],
[fac,oui]]).

classe(zinc,  [[pm,26], [tsech,5], [prom,0], [comp,mono],
[fac,oui]]).

classe(alu,  [[pm,26], [tsech,5], [prom,0], [comp,mono],
[fac,oui]]).

classe(plomb,  [[pm,26], [tsech,5], [prom,0], [comp,mono],
[fac,oui]]).

classe(int,  [[pm,27], [tsech,1], [prom,0], [comp,mono],
[fac,oui]]).

classe(ext,  [[pm,27], [tsech,15], [prom,0], [comp,mono],
[fac,oui]]).

classe(rural,  [[pm,26], [tsech,5], [prom,0], [comp,mono],
[fac,oui]]).

classe(urbain,  [[pm,26], [tsech,5], [prom,0], [comp,mono],
[fac,oui]]).

classe(indust,  [[pm,28], [tsech,15], [prom,0], [comp,mono],
[fac,oui]]).

classe(marin,  [[pm,26], [tsech,1], [prom,0], [comp,mono],
[fac,oui]]).

classe(peuagressif,  [[pm,26], [tsech,1], [prom,0],
[comp,mono], [fac,oui]]).

classe(moyenagressif,  [[pm,26], [tsech,1], [prom,0],
[comp,mono], [fac,oui]]).

classe(tresagressif,  [[pm,36], [tsech,15], [prom,0],
[comp,mono], [fac,oui]]).

classe(jardin,  [[pm,71], [tsech,24], [prom,0], [comp,mono],
[fac,oui]]).

classe(charpente,  [[pm,47], [tsech,240], [prom,0],
[comp,mono], [fac,oui]]).

classe(peusalissant,  [[pm,1000], [tsech,16], [prom,0],
[comp,mono], [fac,oui]]).

classe(antiderapant,  [[pm,56], [tsech,4], [prom,0],
[comp,mono], [fac,oui]]).

classe(primer,  [[pm,45], [tsech,5], [prom,0], [comp,mono],
[fac,oui]]).
```

```
classe(translucide,      [[pm,26],      [tsech,1],      [prom,0],
[comp,mono], [fac,oui]]).

classe(vernis,   [[pm,26],  [tsech,1],  [prom,0], [comp,mono],
[fac,oui]]).

classe(brillant,   [[pm,26], [tsech,1], [prom,0], [comp,mono],
[fac,oui]]).

classe(filmogene,  [[pm,26], [tsech,1], [prom,0], [comp,mono],
[fac,oui]]).

classe(lisse,   [[pm,26],  [tsech,1],  [prom,0],  [comp,mono],
[fac,oui]]).

classe(mat,   [[pm,33],  [tsech,15],  [prom,0],  [comp,mono],
[fac,oui]]).

classe(satine,   [[pm,27],  [tsech,8],  [prom,0], [comp,mono],
[fac,oui]]).

classe(lisse,   [[pm,27],  [tsech,8],  [prom,0],  [comp,mono],
[fac,oui]]).

classe(colore,   [[pm,27],  [tsech,4],  [prom,0], [comp,mono],
[fac,oui]]).

classe(non-filmogene,      [[pm,27],    [tsech,24],    [prom,0],
[comp,mono], [fac,oui]]).

classe(opaque,   [[pm,28],  [tsech,6],  [prom,0], [comp,mono],
[fac,oui]]).

classe(eau,    [[pm,27],    [tsech,4],   [prom,0],   [comp,mono],
[fac,oui]]).

classe(lavage,   [[pm,39],  [tsech,4],  [prom,0], [comp,mono],
[fac,oui]]).

classe(lavagemoyen,      [[pm,27],      [tsech,4],      [prom,0],
[comp,mono], [fac,oui]]).

classe(abrasionforte,      [[pm,56],      [tsech,4],      [prom,0],
[comp,mono], [fac,oui]]).

classe(abrasionmoeynne,      [[pm,26],    [tsech,1],    [prom,0],
[comp,mono], [fac,oui]]).

classe(abrasionlegere,      [[pm,26],      [tsech,1],      [prom,0],
[comp,mono], [fac,oui]]).

classe(solvants, [[pm,56], [tsech,4], [prom,0], [comp,mono],
[fac,oui]]).
```

```
classes(stageau,  [[pm,35],  [tsech,6], [prom,0], [comp,mono],
[fac,oui]]).

classe(bleu,    [[pm,27],  [tsech,4],  [prom,0],  [comp,mono],
[fac,oui]]).

classe(pourri,   [[pm,27],  [tsech,4],  [prom,0], [comp,mono],
[fac,oui]]).

classe(uv,    [[pm,27],   [tsech,4],   [prom,0],  [comp,mono],
[fac,oui]]).

classe(insectes, [[pm,58], [tsech,24], [prom,0], [comp,mono],
[fac,oui]]).

classe(corrosion,  [[pm,26], [tsech,1], [prom,0], [comp,mono],
[fac,oui]]).
[['Cet',etat,comprend,des,caracteristiques,'speciales:',F,Carf,'.'],
[que,le,traitement,ne,possede,'pas','(regle:19).']]).
/* L'EXPLICATION DES SYSTEMES DE PEINTURE.                          */

/* PREMIERE CLASSE DE FAITS                                         */

/* ENFANT                                                           */
enfant(faitavancer,[avancap,avancc,avanca,avancr,avancp]).

enfant(convient,[convientnat,convientforme,convientenv,convientap,
convientcarspec,convientres,convientes,convientas,convientdt]).

enfant(conviendt,[compatair,compatliant,compatsolv,compatpigli,
compatpvc]).

enfant(compatpigli,[ok]).

/* PARENT                                                           */

parent(avanca,faitavancer).

parent(avancap,faitavancer).

parent(avancc,faitavancer).

parent(avancr,faitavancer).

parent(avancp,faitavancer).

parent(convientnat,convient).

parent(convientforme,convient).

parent(convientenv,convient).

parent(convientap,convient).

parent(convientcarspec,convient).
```

```
parent(convientres,convient).

parent(convientes,convient).

parent(convientas,convient).

parent(convientdt,convient).

parent(compatair,convientdt).

parent(compatliant,convientdt).

parent(compatsolv,convientdt).

parent(compatpigli,convientdt).

parent(compatpvc,convientdt).

parent(ok,conpatpvc).

/* DEUXIEME CLASSE DE FAITS                                          */

/*       LES PHRASES                                                 */

francais([faitavancer,2,_],[['fait',avancer,la,'solution.','Regle','n1.'
francais([faitavancer,echec,_],[['ne',fait,pas,'avancer.','Regle','n1.']

francais([avancr,1,[Ress,Resf,A]],
[['Cet',etat,comprend,des,'resistances:',Resf,'.'],

francais([avanca,1,[[Ass,Asf],[As,A]]],
[['Les',aspects,Asf,de,cet,etat,comprennent,des,aspects,a,'remplacer'],
['et',a,'supprimer.','Un',de,ces,aspects,A,est,assure,par,le,traitement]
[puisque,les,aspects,'qu''il',donne,sont,Ass,'(regle:17).']]).

francais([avanca,echec,[[Ass,Asf],[As,A]]],
[['Les',aspects,Asf,de,cet,etat,ne,comprennent,pas,des,aspects,a,'rempla
[ou,a,supprimer,de,la,part,du,'traitement','(regle:17).']]).

francais([convient,echec,_],
[['Le',traitement,ne,'convient',pas,au,'support','(regle:16).']]).

francais([convient,1,_],
[['Le',traitement,'convient',au,'support','(regle:16),']]).

francais([avancap,1,[Aps]],
[['L''etat',comporte,une,apllication,'particuliere:',Aps,et,le,traitemen
[est,adapte,a,cette,application,particuliere,'(regle:18).']]).

francais([avancap,echec,[Aps]],
[['Le',traitement,'n''est',pas,adapte,a,'l''application',particuliere,Ap
[de,'l''etat','(regle:18).']]).

francais([avancc,1,[Cars,F,Carf]],
```

```
[['Cet',etat,comprend,des,caracteristiques,'speciales:',F,Carf,'.'],
['Ces',caracteristiques,sont,toutes,assurees,par,le,traitement],
[puisque,ses,caracteristiques,speciales,sont,Cars,'(regle:19).']]).

francais([avancc,echec,[F,Carf]],
['Une',de,ces,resistances,A,est,assuree,par,le,traitement,puisque,ses],
[resistances,sont,Ress,et,'qu''elles',comportent],
[des,exigences,superieures,a,A,'(regle:20).']]).

francais([avancr,echec,[Ress,Resf]],
[['Cet',etat,comprend,des,'resistances:',Resf,'.'],
['Le',traitement,ne,possede,pas,les,exigences,de,cet,'etat','(regle:20).

francais([avancp,1,[Pros,Prof,A]],
[['Cet',etat,comprend,des,protections,Prof,une,de,ces],
[protections,A,est,assuree,par,le,traitement,puisque],
[ses,protections,sont,Pros,'(regle:21).']]).

francais([avancp,echec,[Pros,Prof]],
[['Les',protections,Pros,'qu''assure',le,traitement,ne,couvrent],
[aucune,des,protections,Prof,specifiees,dans,'l''etat','(regle:21).']]).

francais([convientnat,1,[S]],
[['La',nature,'n''etant',pas,specifiee,dans,'l''etat',',',le,traitement]
[qui,est,appropriе,pour,S,'convient','(regle:2).']]).

francais([convientnat,2,[F]],
[['La',nature,'n''etant',pas,specifiee,dans,le,'traitement'],
['celui-ci',convient,pour,la,nature,particuliere,de,'l''etat',F],
['(regle:2).']]).

francais([convientnat,3,[Natf,N,S]],
[['La',nature,specifiee,dans,'l''etat',etant,Natf,et,les,natures],
[pour,lesquelles,le,traitement,est,appropriе,etant,S,','],
[cela,convient,du,point,de,vue,de,la,'nature','(regle:2).']]).

francais([convientnat,echec,[Natf,N,S]],
[['La',liste,de,natures,S,pour,laquelle,le,traitement,est,appropriе],
[ne,comprend,pas,la,nature,Natf,specifiee,dans,'l''etat','(regle:2).']])

francais([convientforme,1,[S]],
[['La',forme,'n''etant',pas,specifiee,dans,'l''etat,',le,traitement],
[qui,est,appropriе,pour,S,'convient','(regle:3).']]).

francais([convientforme,2,[F]],
[['La',forme,'n''etant',pas,specifiee,dans,le,'traitement,'],
[celui-ci,convient,pour,la,forme,particuliere,F,de,'l''etat','(regle:3).

francais([convientforme,3,[F,S,Forme]],
[['La',forme,F,specifiee,dans,'l''etat',est,moins,exigeant,que,les,forme
[S,Forme,pour,lesquelles,est,appropriе,le],
[traitement,'convient','(regle:3).']]).

francais([convientforme,echec,[F,S,Forme]],
[['Le',traitement,est,appropriе,pour,les,formes,S,Forme,'.'],
```

```
['Il',ne,convient,donc,pas,pour,la,forme,S,specifiee,dans,'l''etat,'],
[puisque,celles-ci,sont,de,types,differents,ou,ont,des],
[exigences,'superieures','(regle:3).']]).

francais([convientenv,1,[_]],
[['L''environnement',de,'l''etat','n''est',pas,'specifie','(regle:4).']]

francais([convientenv,2,[E]],
[['Le',traitement,etant,approprie,pour,tous,les,'environments,'],
['il',convient,pour,'l''environnement',particulier,de],
['l''etat:',E,'(regle:4).']]).

francais([convientenv,3,[E,F,S]],
[['Le',traitement,est,approprie,pour,les,environnements,S,'.'],
['Il',convient,donc,pour,'l''environnement',E,F,specifie,dans,'l''etat']
[puisque,'ceux-ci',ont,des,exigences,egales,ou,'moindres','(regle:4).']]

francais([convientenv,echec,[E,F,S]],
[['Le',traitement,est,approprie,pour,la,liste,'d''environnements'],
[S,'il',ne,convient,pas,pour,la,liste,'d''environnements',E,F],
[specifies,dans,'l''etat,',puisque,ceux,ci,sont,de,types,differents],
[ou,ont,des,exigences,'superieures','(regle:4).']]).


francais([convientap,1,[F]],
[['Le',traitement,est,approprie,pour,toutes,les,applications],
['particulieres.','Il',convient,donc,pour,'l''application'],
[particuliere,F,specifiee,dans,'l''etat','(regle:5).']]).

francais([convientap,2,[X]],
[['Le',traitement,est,approprie,pour,les,applications,'particulieres'],
[X,specifiees,dans,'l''etat','(regle:5).']]).

francais([convientap,3,[F,prepa]],
[['Le',traitement,est,une,'preparation,',il,'n''intervient',donc,pas],
[dans,le,concept,'d''applications','particulieres','(regle:5).']]).

francais([convientap,echec,[X]],
[['Le',traitement,'n''est',approprie,a,aucune,application],
['particuliere.','Il',ne,convient,donc,'pas,','(regle:5).']]).

francais([avancap,echec,[Aps]],
[['Le',traitement,'n''est',pas,adapte,a,'l''application',particuliere,Ap
[de,'l''etat.','(regle:5)']]).

francais([convientcarspec,1,[S]],
[['La',caracteristique,speciale,'n''etant',pas,specifiee,dans,'l''etat']
['le','traitement',qui,est,approprie,pour,S,',','convient','(regle:6).']

francais([convientcarspec,2,[F,S]],
[['Le',traitement,est,approprie,pour,les,caracteristiques,speciales],
[S,'.','Il',convient,donc,pour,les,caracteristiques,speciales,F],
[specifiees,dans,'l''etat','(regle:6).']]).

francais([convientcarspec,echec,[F,S]],
```

```
[['Les',caracteristiques,speciales,S,assurees,par,le,traitement],
[ne,comprennent,pas,toutes,les,caracteristiques,speciales,F],
[specifiees,dans,'l''etat','(regle:6).']]).

francais([convientres,1,[S]],
[['La',resistance,'n''etant',pas,specifiee,dans,'l''etat,'],
['le',traitement,qui,a,pour,resistance,S,',','convient,','(regle:7).']])

francais([convientres,2,[R,F,S]],
[['Le',traitement,est,approprie,pour,les,resistances,S,'.'],
['Il',convient,donc,pour,les,resistances,R,F,specifiees,dans],
['l''etat,',puisque,'celles-ci',ont,des,exigences],
[egales,ou,'moindres','(regle:7).']]).

francais([convientres,echec,[R,F,S]],
[['Le',traitement,est,approprie,pour,les,resistances,S,'.'],
['Il',ne,convient,pas,pour,la,liste,des,resistances,R,F],
[specifiees,dans,'l''etat',puisque,'celles-ci',sont,de,types],
[differents,ou,ont,des,exigences,'superieures','(regle:7).']]).

francais([convientes,1,[F,S]],
[['Le',traitement,est,approprie,pour,les,etats,de,surface,S,'.'],
['Il',convient,donc,pour,les,etats,de,surface,F,specifies],
[dans,'l''etat','(regle:8).']]).

francais([convientes,2,[F,S]],
[['La',liste,'d''etats',de,surface,S],
[que,procure,le,traitement,permet,de,se,rapprocher,de,'l''etat'],
[recouvrable,specifie,dans,'l''etat','(regle:8).']]).

francais([convientes,echec,[recouvrable,S]],
[['L''etat',de,surface,F,procure,par,le,traitement],
[ne,permet,pas,'d''avoir','l''etat',recouvrable],
[specifie,dans,'l''etat','(regle:8).']]).

francais([convientes,echec,[F,S]],
[['La',liste,'d''etats',de,surface,S,que,procure],
[le,traitement,ne,comprend,pas,tous,les,etats,de,surface,F],
[specifies,dans,'l''etat','(regle:8).']]).

francais([convientas,1,[S]],
[['L''aspect','n''etant',pas,specifie,dans,'l''etat,',le,traitement],
[qui,est,approprie,pour,S,',','convient','(regle:22).']]).

francais([convientas,2,[F,S]],
[['Le',traitement,donne,les,aspects,S,'.'],
['Il',convient,'puisqu''il',couvre,les,aspects,permanents,et,a],
[remplacer,qui,sont,compris,dans,les,aspects,F,specifies],
[dans,'l''etat','(regle:22).']]).

francais([convientas,3,[F]],
[['Le',traitement,ne,donne,pas,'d''aspects','particuliers.'],
['Il',convient,puisque,les,aspects,F,specifies,dans,'l''etat',ne],
[comprennent,pas,'d''aspects','permanents','(regle:22).']]).
```

```
francais([convientas,echec,[F,S]],
[['Les',aspects,de,'l''etat',comprend,des,aspects,permanents,et],
[a,'remplacer,','c''est-a-dire',des,aspects,a,satisfaire],
['immediatement.','Or',cette,liste,'d''aspects','n''est',pas],
[comprise,dans,la,liste,des,aspects,S,que,procure],
[le,'traitement','(regle:22).']]).

francais([convientdt,1,[S]],
[['Le',traitement,S,est,la,couche,de,finition,'(regle:9)',et,elle,peut,r
[au,contact,de,'l''air.','(Regle:10)']]).

francais([convientdt,2,[F,[]]],
[[]]).

francais([convientdt,3,[F,Lf,S,Ls]],
[['Le',traitement,F,convient,au,traitement,S,'(regle:9)']]).

francais([conviendt,4,[F,S]],
[['Toute',preparation,en,'l''occurrence',S,peut,se,faire,avant],
['l''application',de,tout,traitement,en,particulier,F,'(regle:9).']]).

francais([convientdt,echec,[[],[S|Ls]]],
[['Le',traitement,est,la,couche,de,'finition,','(regle:9)',or,ce,traitem
[ne,peut,rester,au,contact,de,'l''air.','(regle:10)']])        :-
not([S|Ls] = []).

francais([convientdt,echec,[[F|Lf],[S|Ls]]],
[['Le',traitement,ne,convient,pas,'(regle:9)',car]]).


francais([compatair,1,[S,Pvc]],
[['La',valeur,Pvc,du,'PVC',est,'<=0.40',ce,qui,signifie,que,ce],
['n''est',pas,trop,rugueux,pour,servir,de,couche,de,'finition','(regle:1

francais([compatair,2,[S,Env]],
[['L''environnement',Env,'n''est',pas,trop,agressif,ce,qui],
[signifie,que,ce,traitement,S,peut,servir],
[de,couche,de,'finition','(regle:10).']]).

francais([compatair,echec,[S]],
[['L''environnement',est,trop,agressif,pour,S,'(regle:10).']]).

francais([compatliant,1,[F,S,Lf,Ls]],
[['Les',liants,Lf,de,F,sont,plus,thermoplastes],
[que,les,liants,Ls,de,S,'(regle:11).']]).

francais([compatliant,echec,[F,S,Lf,Ls]],
[['Les',liants,Lf,de,F,sont,moins,thermoplastes],
[que,les,liants,Ls,de,S,'(regle:11).']]).

francais([compatsolv,1,[F,S,Lf,Ls]],
[['Les',solvants,Lf,de,F,sont,moins,agressifs],
[que,les,solvants,Ls,de,S,'(regle:12).']]).

francais([compatsolv,echec,[F,S,Lf,Ls]],
```

```
[['Les',solvants,Lf,de,F,sont,plus,agressifs],
[que,les,solvants,Ls,de,S,'(regle:12).']]).


francais([compatpigli,1,[F,S,Lf,Lp]],
[['Les',pigments,Lp,de,S,et,les,liants,Lf],
[de,F,'(regle:14)','s''accordent']]).

francais([compatpigli,echec,[F,S,Lf,Lp]],
[['(regle:14)','Les',pigments,Lp,de,S,et,les,liants,Lf],
[de,F,ne,peuvent,'s''accorder']]).

francais([compatpvc,1,[F,S,Pf,Ps]],
[['La',valeur,Pf,du,'PVC',de,F,est,superieure,a,la],
[valeur,de,S,'(regle:13).']]).

francais([compatpvc,echec,[F,S,Pf,Ps]],
[['La',valeur,Pf,du,'PVC',de,F,'n''est',pas,superieure,a,la],
[valeur,de,S,'(regle:13).']]).

francais([ok,1,[L]],
[['Le',liant,L,est,'saponifiable','(regle:15).']]).

francais([ok,2,[L]],
[['Il','n''y',a,pas,de,'pigments','(regle:15).']]).

francais([ok,3,[L]],
[['Les',pigments,L,sont,composes,de,'zinc','(regle:15).']]).

francais([ok,echec,[L]],
[['Le',liant,L,'n''est','saponifiable','(regle:15).']]).


/* PHRASE CONCERNANT LA NATURE DU SYSTEME.                          */

expl1niv(Cat,Etat,S,Sysfich,Sysrendu) :-
                          expl1(Etat,Cat,S,Sysfich,Sysrendu,Liste),
                          sortie1niv(Liste).


sortie1niv([]).
sortie1niv([L1|L]) :- sortphr(L1),
                      sortie1niv(L).

expl1(1,1,S,_,_,[['Votre','systeme',S,'est','correct','.']]).

expl1(1,2,S,_,_,[[S,'n''a',pas,ete,repris,pour,des,raisons,economiques,'

expl1(1,3,S,_,_,[[S,'n''est',pas,un,systeme,final,'.'],
['Nous',avons,arrete,son,etude,pour,des,raisons,economiques,'.']]).

expl1(1,4,S,_,_,[Phrase,
[L,fait,avancer,le,systeme,vers,'l''etat',desire,mais,ne,convient],
[pas,techniquement,'.']]) :- enleve1ele(S,S1,L),
                          (S1 = [],A = [];
                           Phrase = [S1,convient,techniquement,'.']).
```

```
expl1(1,5,S,_,_,[[S,convient,techniquement,mais,il,'n''y',a,pas,de,
traitement,qui,font,avancer], [vers,'l''etat',desire,'.']]).

expl1(2,Cat,S,Sysfich,_,[[S,a,trop,peu,de,traitements,'.'],
      [Sysfich,est,deja,plus,complet,'.'], Phrase]) :-
                expl1(1,Cat,Sysfich,_,_,Phrase).

expl1(3,1,S,Sysfich,_,[[S,a,un,ou,des,traitements,de,trop,'.',Sysfich,es
[que,vous,desirez,'.']]).

expl1(3,2,S,Sysfich,_,[[S,a,un,ou,des,traitements,de,trop,'.',Sysfich,es
[que,vous,desirez,mais,'n''est',pas,repris,pour,des,raisons,economiques,

expl1(3,3,S,Sysfich,_,[[S,est,peut,etre,correct,mais,pour,des,raisons,ec
[',',nous,avons,etudie,votre,systeme,'jusque',Sysfich,'.']]).

expl1(3,4,S,Sysfich,_,[['Le','traitement
',L,fait,avancer,le,systeme,partiel,A1],
[mais,L,ne,convient,pas,techniquement,'.'],
[S,'n''est',donc,pas,correct,'.']]) :- enleve(Sysfich,A1,L).

expl1(3,5,S,Sysfich,_,[['Apres',Sysfich,on,ne,peut,pas,appliquer,le,trai
[que,vous,proposez,car,il,ne,fait,pas,avancer,'.']]).


expl1(4,Cat,S,Sysfich,Sysrendu,[['Votre',systeme,S,'n''est',pas,correct,
['Il',est,bon,jusque,Sysrendu],
['On',peut,mettre,des,traitements,de,plus,sur,Sysrendu,'.'],Phrase])
:- expl1(1,Cat,Sysfich,_,_,Phrase).

/* INTERFACE                                                        * /
traduction(1,faitavancer,[faitavancer]).

traduction(2,convientnat,[de,sa,nature]).

traduction(3,convientforme,[de,sa,forme]).

traduction(4,convientenv,[de,son,environnement]).

traduction(5,convientap,[de,son,application,particuliere]).

traduction(6,convientcarspec,[de,ses,caracteristiques,speciales]).

traduction(7,convientres,[de,sa,resistance]).

traduction(8,convientes,[de,son,etat,de,surface]).

traduction(9,convientdt,[de,son,dernier,traitement]).

traduction(10,compatair,[de,sa,compatibilite,a,'l''air']).

traduction(11,compatliant,[de,sa,compatibilite,aux,liants]).

traduction(12,compatsolv,[de,sa,compatibilite,aux,solvants]).
```

```
traduction(13,compatpvc,[de,son,PVC]).

traduction(14,compatpigli,[de,ses,pigments,et,liants]).

traduction(15,ok).

traduction(16,convient).

traduction(17,avanca,[de,son,aspect]).

traduction(18,avancap,[de,son,application,particuliere]).

traduction(19,avancc,[de,sa,caracteristique,speciale]).

traduction(20,avancr,[de,sa,resistance]).

traduction(21,avancp,[de,sa,protection]).

traduction(22,convientas,[de,son,aspect]).


/*            B. BASE DE REGLES                                    */

/*            B.1. PRIMITIVES                                      */

/*            B.1.1. MANIPULATION DE LISTES  : APPARTIENT
                                               CONCAT
                                               DEDOUBLE
                                               DIFF
                                               INCLUS
                                               INFEGAL
                                               INFEGALAUX           */


max([L],L).
max([L1|L2],Max) :- max(L2,M),
                    (L1>M,Max = L1;Max = M).

enlevelele(S,S1,L) :- renvliste(S,[],[L|List]),
                      renvliste(List,[],S1).
egalprod([L1|L],P,L1) :- verifprod(P,L1).
egalprod([_|L],P,L1) :- egalprod(L,P,L1).
appartien(T ,[]).
appartient( T, [T|Q]).
appartient( L, [T|Q]) :- appartient( L, Q).

membre( El, L) :- appartient(El, L).
membre( Nat, L) :- appartient(metaux,L),
                   appartient(Nat, [acier, galva, cuivre, zinc,
                                    alu, plomb]).
equivalent( [recouvrable],L) :-
                   etat_de_depart(Ed),
```

```
                    extnat(Ed,Nat),
                    appartient(bois,Nat),
                    inclus(L,[noeud_non, exudation_non, porosite_non,
                              fissuration_non,biologique_non,bleuissement_
                              cloquage_non ,defaut_non, humidite_non,
                              arete_non,graisse_non,proprete_oui,poussiere
                              !.

equivalent( [recouvrable], L) :-
          etat_de_depart(Ed),
          extnat(Ed,Nat),
          not(appartient(bois,Nat)),
          inclus( L, [corrosion_non, graisse_non, poussiere_non,
                                proprete_oui, accrochage_oui, nettete_nu,
                                nettete_bon_etat]),!.
nombre([],0).
nombre([L1|L2],Y) :- nombre(L2,Y1),
                     Y is Y1 + 1.

tri([],[],_).
tri([X|L],M,C) :- tri(L,N,C),trier(X,N,M,C).
trier(X,[A|L],[A|M],C) :-
           sup(A,X,C),!,
           trier(X,L,M,C).
trier(X,L,[X|L],C).

sup([_,Ecm],[_,Ec],2) :-
                          extev(Ecm,Evm),
                          extev(Ec,Ev),
                          Evm > Ev.

sup(Ecm,Ec,1) :-
                          extev(Ecm,Evm),
                          extev(Ec,Ev),
                          Evm < Ev.

sup(Emax,E1,3) :- Emax > E1.

sup(Ecm,Ec,4) :- extev(Ecm,Evm),
                 extev(Ec,Ev),
                 Evm > Ev.
inclus([],S).
inclus([F|Qf], S) :- appartient(F,S),
                     inclus(Qf,S).

concat([], [], []).
concat([A|La], [], [A|Lc]) :- concat(La, [], Lc).
concat([A|La], [A|Lb], C) :- concat(La, Lb, C).

dedouble( [], []).
dedouble( La, [B|Lb]) :- appartient( B, Lb),!,
                         dedouble( La, Lb).
dedouble([A|La], [A|Lb]) :- dedouble( La, Lb).

/* diff( A,B,C) => C= A\B au sens mathematique avec eventuellement des d
```

```
diff([], B, []).
diff([A|La], B, Lc) :- appartient( A, B),!,
                       diff( La, B, Lc).
diff([A|La], B, [A|Lc]) :- diff(La, B, Lc).


infegal(A,A,L) :- appartient(A,L).
infegal(A,B,[A|L]) :- appartient(B,L).
infegal(A,B,[T|Q]) :- infegal(A,B,Q).

infegalaux(El,[Comp|C],L) :- infegal(El,Comp,L);
                             infegalaux(El,C,L).

/*                  MANIPULATION DE FICHIER                              */


lectfich(Nonfich,A) :- see(Nonfich),lire(A),seen.

lire(A) :- read(B),
           ( B = end_of_file,A = [],!;
             A = [B|C], lire(C)).

ecrirefich([]).
ecrirefich([L1|L]) :- write(L1), write('.'),nl,
                      ecrirefich(L).


/*            MANIPULATION DE LISTE_DE_CRITERES  : VALEUR_ITEM
                                                   EXTVALCRIT */

valeur_item([Item, Valeur], [], Res) :- Res is 2, !.

valeur_item([Item, Valeur], [[Item, Valeur]|L], Res) :- Res is 1, !.

valeur_item([Item, Valeur], [[Item,Va]|L], Res) :- Res is 0, !.

valeur_item([Item, Valeur], [L|L1], Res) :-
                        valeur_item([Item, Valeur], L1, Res).

/*          B.1.3. PRIMITIVES D'EXTRACTION : EXTNAT
                                             EXTFORME
                                             ...                      */

extpvc(Trt, Pvc) :- comp(Trt, C),
                    valeur_item([pvc, Pvc], C, 1).

extliant(Trt, Liant) :- comp(Trt, C),
                        valeur_item([l, Liant], C, 1).

extsol(Trt, Sol) :- comp(Trt, C),
                    valeur_item([s, Sol], C, 1).

extpig(Trt, Pig) :- comp(Trt, C),
                    valeur_item([p, Pig], C, 1).
```

```
extvalcrit(Nomcrit, L, Valcrit) :- valeur_item([Nomcrit, Valcrit], L, 1)
                                   !.
extvalcrit(Nomcrit, L, []) :- valeur_item([Nomcrit, Valcrit], L, 2),!.

extnat(Etat, Nature) :- extvalcrit(nature, Etat, Nature).
extforme(Etat, Forme) :- extvalcrit(forme, Etat, Forme).
extenv(Etat, Env) :- extvalcrit(env, Etat, Env).
extap(Etat, Ap) :- extvalcrit(applipart, Etat, Ap).
extcarspec(Etat, Carspec) :- extvalcrit(carspec, Etat, Carspec).
extetatsurf(Etat, Etatsurf) :- extvalcrit(etatsurf, Etat, Etatsurf).
extas(Etat, Aspect) :- extvalcrit(aspect, Etat, Aspect).
extres(Etat, Resist) :- extvalcrit(resist, Etat, Resist).
extprot(Etat, Prot) :- extvalcrit(protect, Etat, Prot).
extderntrt(Etat, Dt) :- extvalcrit(derntrt, Etat, Dt).
extev(Etat,Ev) :- extvalcrit(evalu, Etat, Ev).
extprom(Etat,Pro) :- extvalcrit(prom, Etat, Pro).
extfac(Etat,Fac) :- extvalcrit(fac, Etat, Fac).
extnpro(Etat,Np) :- extvalcrit(npro, Etat, Np).
extodeur(Etat, Od) :- extvalcrit(odeur, Etat, Od).
extanc(Etat, Anc) :- extvalcrit(anci, Etat, Anc).
extpist(Etat, Pis) :- extvalcrit(maplip, Etat, Pis).
extroul(Etat, Rou) :- extvalcrit(maplir, Etat, Rou).
extprix(Etat, Pr) :- extvalcrit(pm, Etat, Pr).
exttsech(Etat, Ts) :- extvalcrit(tsech, Etat, Ts).
extcomp(Etat, Com) :- extvalcrit(comp, Etat, Com).
extqual(Etat,Qua) :- extvalcrit(qual, Etat, Qua).
extsol(Etat, So) :- extvalcrit(sol, Etat, So).



extitemsvect([[nature, Nat], [forme, Forme], [env, Env],
                [applipart, Ap], [carspec, Car],
                [etatsurf, Es], [aspect, As], [resist, Res], [protect,
                [derntrt, Dt]], Nat, Forme, Env, Ap, Car, Es, As, Res,

extitemsvect( V, Nat, Forme, Env, Ap, Car, Es, As, Res, Pro, Dt) :-
                 extnat( V, Nat),
                 extforme( V, Forme),
                 extenv( V, Env),
                 extap( V, Ap),
                 extcarspec( V, Car),
                 extetatsurf( V, Es),
                 extas( V, As),
                 extres( V, Res),
                 extprot( V, Pro),
                 extderntrt( V, Dt).

extlistitem( Type, Item, [], []) :- !.
extlistitem( Type, Item, L, []) :- not(itemvalclasse(Item, S, Type)), !.
extlistitem( Type, Item, [I|Linit], R) :-
                 itemvalclasse( Item, S, Type),
                 extlistaux( [I|Linit], R, S), !.

extlistaux( Init, [], []).
```

```
extlistaux([], [], S).
extlistaux([I|Linit], [I|Lres], S) :-
                    appartient( I, S),
                    extlistaux(Linit, Lres, S).
extlistaux([I|Linit], Lres, S) :-
                    extlistaux(Linit, Lres, S).

/*          PROCEDURE DE MANIPULATION DE LA BASE DE FAITS          */

lirequestion(Q) :- lirequest(Q,[]).

lirequest([Q1|Q2],S) :- question(Q1),
                        not(appartient(Q1,S)),
                        lirequest(Q2,[Q1|S]).
lirequest([],_).


liresol(L) :- lireso(L,[]),!.

lireso([L1|L2],S) :- sol(L1),
                     not(appartient(L1,S)),
                     lireso(L2,[L1|S]).
lireso([],_).

liresolterm(L) :- liresolter(L,[]),!.

liresolter([L1|L2],S) :- solterm(L1),
                         not(appartient(L1,S)),
                         liresolter(L2,[L1|S]).
liresolter([],_).

lectsolu([L1|L2]) :- retract(solu(L1)),
                     lectsolu(L2).
lectsolu([]).


/*              C. INTERFACE                                       */

/*              C.1. OUTILS POUR L'INTERFACE                       */

/* 1. Sortie d'une phrase à l'écran*/

sortphr([]) :- nl.

sortphr([T|Q]) :- write(T),
                  tab(1),
                  sortphr(Q).


/* 2. Inversion de l'ordre des éléments d'une liste*/

renvliste([],L,L).

renvliste([T|Q], L, R) :- renvliste(Q, [T|L], R).
```

```
/* 3. Sortie d'une liste à l'écran*/

sortliste([]) :- nl, nl.

sortliste([T|Q]) :- write(T),
                    nl,
                    sortliste(Q).


/* 4. Formation de l'état de surface à partir de ses valeurs */

accouple( Gr, Pous, Prop, [ G, Po, Pr]) :-
                couppleg(Gr, G),
                couplepo(Pous, Po),
                couplepr(Prop, Pr).

accouplemetaux( Co, Ac, Ne, [ C, A, N]) :-
                coupleco(Co, C),
                coupleac(Ac, A),
                couplene(Ne, N).

accouplebois( No, Ex, Po, Fi, Bi, Bl, Cl,  De, Hu, Ar, [ N, E, P,
F, B, Bleu, C, D, H, A]) :-
                coupleno(No,N),
                coupleex(Ex,E),
                couplepor(Po,P),
                couplefi(Fi,F),
                couplebi(Bi,B),
                couplebl(Bl,Bleu),
                couplecl(Cl,C),
                couplede(De,D),
                couplehu(Hu,H),
                couplear(Ar,A).



/* 5. Couplage d'un critère d'état de surface et de sa valeur */


coupleg(o, graisse_oui).
coupleg(n, graisse_non).

couplepo(o, poussiere_oui).
couplepo(n, poussiere_non).

couplepr(o, proprete_oui).
couplepr(n, proprete_non).

coupleco(o, corrosion_oui).
coupleco(n, corrosion_non).

coupleac(o, accrochage_oui).
coupleac(n, accrochage_non).
```

```
coupleac(douteux, accrochage_douteux).

couplene(nu, nettete_nu).
couplene(bon_etat, nettete_bon_etat).

coupleno(o,noeud_oui).
coupleno(n,noeud_non).

coupleex(o, excudation_oui).
coupleex(n,excudation_non).


couplepor(o,porosite_oui).
couplepor(n,porosite_non).


couplefi(o,fissuration_oui).
couplefi(n,fissuration_non).

couplebi(n,biologique_non).
couplebi(faible,biologique_faible).
couplebi(fort,biologique_fort).

couplebl(o,bleuissement_oui).
couplebl(n,bleuissement_non).

couplede(o,defaut_oui).
couplede(n,defaut_non).

couplehu(o,humidite_oui).
couplehu(n,humidite_non).

couplear(o,arete_oui).
couplear(n,arete_non).

couplecl(fort,cloquage_fort).
couplecl(n,cloquage_non).
couplecl(faible,cloquage_faible).



/* 1. Petite entête*/

entete :- sortphr(['Bonjour.', 'Veuillez', entrer, successivement]),
          sortphr(['l''etat', initial, et, 'l''etat', desire]),
          sortphr([du, support, apres, 'traitement.']).

/* etat de surface */

etatsurface(Nat , X) :-
                appartient(bois,Nat ),
                tab(5), write('noeud : '),read(No),nl,
                tab(5), write('excudation : '),read(Ex),nl,
                tab(5), write('porosite : '),read(Po),nl,
```

```
                tab(5), write('fissuration : '),read(Fi),nl,
                tab(5), write('biologique : '),read(Bi),nl,
                tab(5), write('bleuissement : '),read(Bl),nl,
                tab(5), write('cloquage : '),read(Cl),nl,
                tab(5), write('defaut : '),read(De),nl,
                tab(5), write('humidite : '),read(Hu),nl,
                tab(5), write('arete : '),read(Ar),nl,
                accouplebois(No,Ex,Po,Fi,Bi,Bl,Cl,De,Hu,Ar,X).

etatsurface(Nat,X) :-
                tab(5), write('corrosion : '), read(Co) , nl,
                tab(5), write('accrochage : '), read(Ac) , nl,
                tab(5), write('nettete : '), read(Ne) , nl,
                accouplemetaux(Co,Ac,Ne,X).
/* 2. Entrée des données*/

entetatinit(Nat, Forme, Env) :-
                sortphr(['Veuillez', decrire, 'l''etat', initial, '.']),
                sortphr(['Pour', les, valeurs, 'possibles,',
                        'reportez-vous', au, mode, 'd''emploi', joint]),
                sortphr([au, programme]),
                nl,
                write('Nature : '), read(Nat), nl,
                write('Forme : '), read(Forme), nl,
                write('Environnement : '), read(Env), nl,
                write('Etat'), tab(1), write(de), tab(1), write('surface
                nl,
                tab(5), write('graisse : '), read(Gr), nl,
                tab(5), write('poussiere : '), read(Pous), nl,
                tab(5), write('proprete : '), read(Prop), nl,
                etatsurface( Nat, X),
                write('Protections : '), read(Prot), nl,
                accouple(Gr,Pous,Prop,E),
                concat(Etsurf,E,X),
                assert(etat_de_depart1( [[nature, Nat],
                                         [forme, Forme],
                                         [env, Env],
                                         [applipart, []],
                                         [carspec, []],
                                         [etatsurf, Etsurf],
                                         [aspect, []],
                                         [resist, []],
                                         [protect, Prot],
                                         [derntrt, []]
                                        ])),
                assert(etat_de_depart2( [[nature, Nat],
                                         [forme, Forme],
                                         [env, Env],
                                         [applipart, []],
                                         [carspec, []],
                                         [etatsurf, [recouvrable]],
                                         [aspect, []],
                                         [resist, []],
                                         [protect, Prot],
                                         [derntrt, []]
```

```
                                                ])).

entetatfin(Nat, Forme, Env) :-
                    sortphr(['Veuillez', decrire, 'l''etat', 'final.']),
                    sortphr(['Pour', les, valeurs, 'possibles,',
                             'reportez-vous', au, mode, 'd''emploi', joint])
                    sortphr([au, programme]),
                    nl,
                    sortphr(['Application', 'particuliere :']), read(Ap), nl
                    write('Aspect : '), read(Asp), nl,
                    write('Caracteristiques'), tab(1), write('speciales : ')
                        read(Carspec), nl,
                    write('Resistances : '), read(Resist), nl,
                    write('Protections : '), read(Protfin), nl,
                    assert(etat_final_com([[sol,[]],
                                           [pm,0],
                                           [comp,0],
                                           [prom,0],
                                           [tsech,0],
                                           [qual,[]],
                                           [anci,a],
                                           [fac,o],
                                           [maplir,n],
                                           [maplip,n],
                                           [odeur,n],
                                           [npro,0],
                                           [evalu,0]])),
                              assert(etat_final1( [[nature, Nat],
                                                   [forme, Forme],
                                                   [env, Env],
                                                   [applipart, []],
                                                   [carspec, []],
                                                   [etatsurf, [recouvrable]],
                                                   [aspect, []],
                                                   [resist, []],
                                                   [protect, []],
                                                   [derntrt, []]
                                                  ])),
                              assert(etat_final2( [[nature, Nat],
                                                   [forme, Forme],
                                                   [env, Env],
                                                   [applipart, Ap],
                                                   [carspec, Carspec],
                                                   [etatsurf, [recouvrable]],
                                                   [aspect, Asp],
                                                   [resist, Resist],
                                                   [protect, Protfin],
                                                   [derntrt, []]
                                                  ])).

entdonnees :- entetatinit(Nat, Forme, Env),
              entetatfin(Nat, Forme, Env).


/* 3. Sortie des résultats*/
```

```
sortresult([]) :- sortphr(['Il', 'n''existe', pas, de, solution, a, votr
                          'probleme.']).


sortresult(Result) :- renvliste( Result, [], Res),
                      sortphr(['La', suite, de, traitements, a, applique
                               'est : ']),
                      nl,
                      sortliste(Res).


sort([]).
sort([Re|L]) :- renvliste(Re,[],Res),
                sortliste(Res),
                nl,
                sort(L).

suit([]).
suit([[]|Lis]) :- suit(Lis).
suit([[A,B,C]|Lis]) :- traduction(_,A,Phr),
                       sotphr(Phr),nl,
                       suit(Lis).

menu :-
   sortphr(['         ','Systeme',expert,en,'peinture.']),
   sortphr(['         ','--------------------------']),nl,nl,
   sortphr(['1:',nouveau,'probleme.']),nl,
   sortphr(['2:',une,autre,'solution.']),nl,
   sortphr(['3:',il,'n''y',a,pas,de,'solution.']),nl,
   sortphr(['4:',plus,de,precision,sur,le,systeme,vu,'precedemment.']),n
   sortphr(['5:','Pourquoi?']),nl,
   sortphr(['6:','Comment?']),nl,
   sortphr(['7:',vous,desirez,utiliser,un,'produit.']),nl,
   sortphr(['8:',explication,'economique.']),nl,
   sortphr(['9:','Fini.']),nl,
   sortphr(['Faite',votre,'choix.']),nl.


sortie([]) :- sortresult([]).
sortie(L) :- soti(L).
soti([]).
soti([L1|L]) :- sortresult(L1),
                soti(L).
sortieprepa([]).
sortieprepa(Pre) :- sortphr(['Avant',de,peindre,il,'faut:']),
                    sortliste(Pre).
menucom(Rep) :- sortphr(['Desirez-vous',plus,'d''explication.']),
                sortphr(['Si',oui,tapez,'1.',si,non,un,autre,'caractere.
                read(Rep).
explicom([L1|Sui]) :-
   effetdebord(L1,[[pm,P],[tsech,[T1,T2]],[comp,Co],[base,Ba],[prom,Pro]
               [fac,Fac],  [stock,_],  [odeur,Od],  [anci,A],
               [maplir,M], [maplip,Mp], [qual,Qua]]),
   impreseffetdebord(L1,P,T1,T2,Co,Ba,Pro,Fac,Od,A,M,Mp,Qua),
```

```
    commer(Sui).

commer([]).
commer([Sui|Suite]) :-
    effetdebord(Sui,[[pm,P],[tsech,[T1,T2]],[comp,Co],[base,Ba],[prom,Pro
                    [fac,Fac],   [stock,_],  [odeur,Od],  [anci,A],
                    [maplir,M], [maplip,Mp], [qual,Qua]]),
    impreseffetdebord(Sui,P,T1,T2,Co,Ba,Pro,Fac,Od,A,M,Mp,Qua),
    commer(Suite).


impreseffetdebord(S,P,T1,T2,Co,Ba,Pro,Fac,Od,A,M,Mp,Qua) :-
    sortphr(['Les',caracteristiques,du,produit,S,'sont:']),nl,
    sortphr([le,prix,au,metre,carre,':',P]),nl,
    sortphr([le,temps,de,sechage,de,la,derniere,couche,'est:',T1]),nl,
    sortphr([le,temps,de,sechage,de,la,premiere,couche,'est:',T2]),nl,
    sortphr(['La',composition,'est:',Co]),nl,
    sortphr(['La',base,'est:',Ba]),nl,
    sortphr(['La',facilite,'d''application','est:',Fac]),nl,
    sortphr(['Le',produit,a,une,'odeur:',Od]),nl,
    sortphr(['Le',produit,est,'connu:',A]),nl,
    sortphr(['Le',produit,est,applicable,au,'rouleau:',M]),nl,
    sortphr(['Le',produit,est,applicable,au,'pistollet:',Mp]),nl,
    sortphr(['La',qualite,du,produit,'est:',Qua]),nl.

expl2com(Cat,Sys) :-
    lectfich(trace,Trace),
    enlevelele(Trace,Rtrace,Etat),
    (Cat = 3,
     recherche(somme,Rtrace,[_,_,Donnee]),
     sortphr(['nous',avons,estime,un,produit,a,appliquer,a,la,suite]),
     sortphr([nous,n'avons,pas,toutes,les,caracteristiques,de,ce,produit]
     donnee = [[_],[Pm,Com,Pom,Tsm,Fm],[P,Co,Pro,T,Fac,Npr]],
     impreseffetdebord([estime],Pm,Tsm,Tsm,Com,0,Pom,Fm,0,0,0,0,0),
     Etat = [_,[[sol,_],[pm,_],[comp,_],[prom,_],[tsech,_],
           [qual,_],[anci,_],[fac,_],[maplir,M],[maplip,Mp],
           [odeur,O],[npro,_],[evalu,Ev]]];
     Etat = [_,[[sol,_],[pm,P],[comp,Co],[prom,Pro],[tsech,T],
           [qual,Qua],[anci,A],[fac,Fac],[maplir,M],[maplip,Mp],
           [odeur,O],[npro,Npr],[evalu,Ev]]]),
    (Npr > 2 ,
     sortphr(['La',somme,des,prix,des,produits,'est',egal,au,prix,du]),nl
     sortphr([systeme,au,metre,'carre:',P]),nl,
     sortphr(['Le',temps,de,sechage,est,egal,a,la,somme,des,temps]),nl,
     sortphr([de,sechage,des,produits,':',T]),nl;
    Npr = 2 ,Sys =[P1,P2],
     sortphr(['Le',prix,du,systeme,est,egal,a]),nl,
     sortphr(['2',*,'Prix(',P1,')',+,'Prix(',P2,')','=',P]),nl,
     sortphr(['Le',temps,de,sechage,du,systeme,est,egal,a]),nl,
     sortphr(['2',*,'Tps(',P1,')','+','Tps(',P2,')','=',T]),nl;
     sortphr(['Le',prix,du,systeme,est,egal,a]),nl,
     sortphr(['3',*,'Prix(',Sys,')','=',P]),nl,
     sortphr(['Le',temps,de,sechage,du,systeme,est,egal,a]),nl,
     sortphr(['3',*,'Tps(',Sys,')','=',T]),nl),
     (Co =bi,
```

```
     sortphr(['Il',existe,un,produit,dont,sa,composition,est,'bicomposan
     nl;
     sortphr(['tout',les,produits,du,systeme,sont,'monocomposants.']),nl
    (Qual = s,
     sortphr(['La',qualite,du,systeme,est,tres,'bonne.']),
     sortphr(['Cela',depend,du,dernier,produit,'appliquer.']),nl;
     sortphr(['La',qualite,du,systeme,est,'bonne.']),
     sortphr(['Cela',depend,du,dernier,produit,'appliquer.']),nl),
    (O = o,
     sortphr(['Il',y,a,au,moins,un,produit,qui,a,une,odeur]);
     sortphr(['Le',systeme,ne,sent,'pas.']),nl),
    (M =o,
     sortphr(['On',peut,utiliser,le,rouleau,pour,au,moins,un,'produit.']
     nl;true),
    (Mp = o,
     sortphr(['On',peut,utiliser,le,pistollet,pour,au,moins,un,'produit.
     nl;true),
    sortphr(['Pour',calculer,la,fonction,'d''evaluation',nous,nous,somme
    nl,
    sortphr([servis,des,coefficiants,'d''importance',et,de,'ponderation.
    nl,
    sortphr(['La',fonction,'d''evaluation',est,egal,'a:',Ev]).


attendre :- read(A).

poser(Ph,Re) :-
        sortphr(Ph),
        nl,
        read(Re).

sortiefam([L1|L]) :- francais(L1,Phrase),
                     sortie1niv(Phrase),nl,
                     sortphr(['car']),nl,
                     sortienf(L).

sortienf([]) :- sortphr(['Tapez',un,caractre,pour,'continuer.']),!,
                read(A).

sortienf([[]|L]) :- sortienf(L).

sortienf([L1|L]) :- francais(L1,Phrase),
                    sortie1niv(Phrase),nl,
                    sortienf(L).

questionregletrt(Regle,Trt) :-
                    sortphr(['Numero',de,la,regle,a,'expliquer.']),
                    read(Regle),
                    sortphr(['Traitement',sur,lequel,porte,la,'regle.'])
                    read(Trt).

sortiepar([]) :- lectfich(trace,Trace),pointeur(_,Syst),
                 expli2niv(Syst,Trace).

sortiepar(Ele) :- sortphr(['Le',parent,de,la,regle,'est:']),
```

```
                    nl,
                    francais(Ele,Phrase),
                    sortie1niv(Phrase).


presentationetat([[[nature,Nat],[forme,For],[env,Env],[applipart,Apl],
[carspec,Car],[etatsurf,Et],[aspect,Asp],[resist,Res],[protect,Pro],
[derntrt,Der]],_]) :-
                     sortphr(['La',nature,du,support,est,Nat,'.']),
                     sortphr(['L''environment',est,Env,'.']),
                     (Apl = [];
                      sortphr(['L''application',particuliere,est,Apl,'.']
                    (Et = [];
                      sortphr(['L''etat',de,surface,est,Et,'.'])),
                    (Res = [];
                      sortphr(['La',resistance,du,support,est,Res,'.'])),
                    (Asp = [];
                      sortphr(['L''aspect',du,support,est,Asp,'.'])),
                    (Pro = [];
                      sortphr(['La',protection,du,support,est,Apl,'.'])).


/*    LES PROCEDURES DE HAUT NIVEAU
/*              REGLES DU MOTEUR D'INFERENCE                          */
                                                                      */
/*            B.2.1. CHOIX DU TRAITEMENT                              */

optimise(Ed) :- extetasurf( Ed, Es),
                equivalent([recouvrable], Es),
                assert(special(pas_prepa)).

optimise(Ed).

enleve :- special(pas_prepa),
          retract(special(S)).
enleve.

convient(L1, Es, Ef) :- extitemsvect( Es, Nats, Formes, Envs, Aps,
                                      Cars, Ess, Ass, Ress, Pros, Dts)
                extitemsvect( Ef, Natf, Formef, Envf, Apf, Carf, Esf, As
                              Resf, Prof, Dtf),

                faitavancer( Aps, Apf, Cars, Carf, Ass, Asf, Ress, Resf,
                             Pros, Prof),

                convientnat(Natf, Nats),
                convientforme(Formef, Formes),
                convientenv(Envf, Envs),
                convientap(Apf, Aps, Cars),
                convientcarspec(Carf, Cars),
                convientas(Asf, Ass),
                convientes(Esf, Ess),
                convientres(Resf, Ress),
                convientdt(Dtf, [L1]),
                (imprime(1),write([convient,1,[L1]]),
```

```
                write('.'),nl;true).

convient(L1,_,_) :-
                (imprime(1),write([convient,echec,[L1]]),
                write('.'),nl,false;false).


faitavancer( _,_,[prepa], _,_,_,_,_,_,_) :- not(special(pas_prepa)).
faitavancer( Aps, Apf, Cars, Carf, Ass, Asf, Ress, Resf, Pros, Prof) :-
                (avancap( Aps, Apf);
                avancc( Cars, Carf);
                avanca( Ass, Asf);
                avancr( Ress, Resf);
                avancp( Pros, Prof)),
                (imprime(1),write([faitavancer,2,[0]]),
                write('.'),nl;true).

faitavancer(_,_,_,_,_,_,_,_,_,_) :-
                (imprime(1),write([faitavancer,echec,[0]]),
                write('.'),nl,false;false).

avancap( Aps, Apf) :- not (concat(Apf, [], [])),
                      diff( Apf, Aps,[]),
                     (imprime(1),write([avancap,1,[Aps]]),
                      write('.'),nl;true).

avancap(Aps,_) :-
                (imprime(1),write([avancap,echec,[Aps]]),
                write('.'),nl,false;false).

avancc( Cars, [F|Carf]) :- inclus( [F|Carf], Cars),
                          (imprime(1),write([avancc,1,[Cars,F,Carf]]),
                           write('.'),nl;true).

avancc(Cars,[F|Carf]) :-
                (imprime(1),write([avancc,echec,[Cars,F,Carf]]),
                 write('.'),nl,false;false).

avanca(Ass, Asf) :- extlistitem(remp, aspect, Asf, R),
                    extlistitem(supp, aspect, Asf, S),
                    concat( As, R, S),
                    appartient( A, As),
                    appartient( A, Ass),
                    (imprime(1),write([avanca,1,[[Ass,Asf],[As,A]]]),
                    write('.'),nl;true).

avanca(Ass,Asf) :- (imprime(1),
                   write([avanca,echec,[[Ass,Asf],[_,_]]]),
                   write('.'),nl,false;false).

avancr(Ress, Resf) :- croiss(resist, L),
                      appartient(A, Resf),
                      infegalaux( A, Ress, L),
                      (imprime(1),write([avancr,1,[Ress,Resf,A]]),
                       write('.'),nl;true).
```

```
avancr(Ress,Resf) :-
                (imprime(1),write([avancr,echec,[Ress,Resf]]),
                 write('.'),nl,false;false).

avancp(Pros, Prof) :- appartient(A, Prof),
                      appartient(A, Pros),
                      (imprime(1),write([avancp,1,[Pros,Prof,A]]),
                       write('.'),nl;true).

avancp(Pros,Prof) :-
                (imprime(1),write([avancp,echec,[Pros,Prof]]),
                 write('.'),nl,false;false).


convientnat([], S) :-
                      (imprime(1),write([convientnat,1,[S]]),
                       write('.'),nl;true).

convientnat(F, []) :-
                      (imprime(1),write([convientnat,2,[F]]),
                       write('.'),nl;true).

convientnat([Natf|N],S) :- membre(Natf, S),
                     (imprime(1),write([convientnat,3,[Natf,N,S]]),
                      write('.'),nl;true).
convientnat([Natf|N],S) :-
                (imprime(1),write([convientnat,echec,[Natf,N,S]]),
                 write('.'),nl,false;false).

convientforme([], S) :-
                      (imprime(1),write([convientforme,1,[S]]),
                       write('.'),nl;true).
convientforme(F, []) :-
                      (imprime(1),write([convientforme,2,[F]]),
                       write('.'),nl;true).

convientforme(F, [S|Formes]) :- croiss( forme, L),
                                not(infegalaux( S, F, L)),
                                convientforme( F,Formes),
                                (imprime(1),write([convientforme,3,[F,S,
                                write('.'),nl;true).
convientforme(F,[S|Forme]) :-
                (imprime(1),write([convientforme,echec,[F,S,Forme]]),
                 write('.'),nl,false;false).

convientenv([], S) :-
                     (imprime(1),write([convientenv,1,[S]]),
                      write('.'),nl;true).
convientenv(E, []) :-
                     (imprime(1),write([convientenv,2,[E]]),
                      write('.'),nl;true).
convientenv([E|F], S) :- croiss( env, L),
                         infegalaux( E,S,L),
                         (imprime(1),write([convientenv,3,[E,F,S]]),
```

```
                        write('.'),nl;e).

convientenv([E|F],S) :-
                (imprime(1),write([convientenv,echec,[E,F,S]]),
                 write('.'),nl,false;false).


convientap(F, [], _) :-
                     extlistitem( perm, applipart, F, Perm),
                     diff( F, Perm, [X|Y]),
                     (imprime(1),write([convientap,1,[F]]),
                      write('.'),nl;true).

convientap(X, X, _) :-
                      (imprime(1),write([convientap,2,[X]]),
                       write('.'),nl;true).

convientap(F, [], [prepa]) :-
                      (imprime(1),write([convientap,3,[F,prepa]]),
                       write('.'),nl;true).
convientap(X,_,_) :-
                (imprime(1),write([convientap,echec,[X]]),
                 write('.'),nl,false;false).


convientcarspec( [], S) :-
                      (imprime(1),write([convientcarspec,1,[S]]),
                       write('.'),nl;true).
convientcarspec( F, S) :- inclus( F, S),
                      (imprime(1),write([convientcarspec,2,[F,S]]),
                       write('.'),nl;true).
convientcarspec(F,S) :-
                (imprime(1),write([convientcarspec,echec,[F,S]]),
                 write('.'),nl,false;false).


convientres([], S) :-
                     (imprime(1),write([convientres,1,[S]]),
                      write('.'),nl;true).
convientres([R|F], S) :- croiss(resist, L),
                         infegalaux(R, S, L),
                         convientres( F, S),
                         (imprime(1),write([convientres,2,[R,F,S]]),
                          write('.'),nl;true).
convientres([R|F],S) :-
                (imprime(1),write([convientres,echec,[R,F,S]]),
                 write('.'),nl,false;false).

convientes(F, S) :- inclus(F, S),!,
                     (imprime(1),write([convientes,1,[F,S]]),
                      write('.'),nl;true).
convientes(F, S) :- equivalent(F, S),!,
                     (imprime(1),write([convientes,2,[F,S]]),
                      write('.'),nl;true).
```

```
convientes(Pros,Prof) :-
                (imprime(1),write([convientes,echec,[Pros,Prof]]),
                 write('.'),nl,false;false).

convientas([], S) :-
                     (imprime(1),write([convientas,1,[S]]),
                      write('.'),nl;true).
convientas(F, S) :- extlistitem( perm, aspect, F, Perm),
                    extlistitem( remp, aspect, F, Remp),
                    inclus( Remp, S),
                    inclus( Perm, S),
                      (imprime(1),write([convientas,2,[F,S]]),
                       write('.'),nl;true).

convientas(F, []) :- extlistitem( perm, aspect, F, Perm),
                     diff( F, Perm, []),
                      (imprime(1),write([convientas,3,[F]]),
                       write('.'),nl;true).

convientas(F,S) :-
                (imprime(1),write([convientas,echec,[F,S]]),
                 write('.'),nl,false;false).


convientdt( [], [S|Ls]) :- compatair(S),
                     (imprime(1),write([convientdt,1,[S]]),
                      write('.'),nl;true).
convientdt( F, []) :- !,
                     (imprime(1),write([convientdt,2,[F,[]]]),
                      write('.'),nl;true).
convientdt( [F|Lf], [S|Ls]) :- compatliant( F, S),
                               compatsolv( F, S),
                               compatpigli( F, S),
                               compatpvc( F, S), !,
                     (imprime(1),write([convientdt,3,[F,Lf,S,Ls]]),
                      write('.'),nl;true).
convientdt( [F], [S]) :- comp(F, _),
                         not(comp(S, _)),
                     (imprime(1),write([convientdt,4,[F,S]]),
                      write('.'),nl;true).
convientdt(F,S) :-
                (imprime(1),write([convientdt,echec,[F,S]]),
                 write('.'),nl,false;false).


compatair(S) :- extpvc(S,Pvc),
                0.40 >= Pvc,
                (imprime(1),write([compatair,1,[S,Pvc]]),
                write('.'),nl;true);
                etat_de_depart(Ed),
                extenv(Ed, Env),
                inclus(Env, [int,peuagressif,moyenagressif,
                                    tresagressif]),
                (imprime(1),write([compatair,2,[S,Env]]),
                write('.'),nl;true).
```

```
compatair(S) :-
                (imprime(1),write([compatair,echec,[S]]),
                 write('.'),nl,false;false).


compatliant( F, S) :- extliant( F, Lf),
                      extliant( S, Ls),
                      echeltherm(A),
                      valeur_item( [Lf, Nf], A, 1),
                      valeur_item( [Ls, Ns], A, 1),
                      Nf =< Ns,
                      (imprime(1),write([compatliant,1,[F,S,Lf,Ls]]),
                       write('.'),nl;true).

compatliant(F,S) :- extliant( F, Lf),
                    extliant( S, Ls),
                     (imprime(1),write([compatliant,echec,[F,S,Lf,Ls]]),
                      write('.'),nl,false;false).


compatsolv( F, S) :- extsol( F, Lf),
                     extsol( S, Ls),
                     echelagr(A),
                     valeur_item( [Lf, Nf], A, 1),
                     valeur_item( [Ls, Ns], A, 1),
                     Nf =< Ns,
                      (imprime(1),write([compatsolv,1,[F,S,Lf,Ls]]),
                       write('.'),nl;true).
avancp(Pros,Prof) :-
                extsol( F, Lf),
                extsol( S, Ls),
                (imprime(1),write([compatsolv,echec,[F,S,Lf,Ls]]),
                 write('.'),nl,false;false).

compatpvc(F, S) :- extpvc(F, Pf),
                   extpvc(S, Ps),
                   Pf =< Ps,
                   (imprime(1),write([compatpvc,1,[F,S,Pf,Ps]]),
                   write('.'),nl;true).

compatpvc(F,S) :- extpvc(F, Pf),
                  extpvc(S, Ps),
                  (imprime(1),write([compatpvc,echec,[F,S,Pf,Ps]]),
                  write('.'),nl,false;false).


compatpigli(F, S) :- extpig( S, Lp),
                     extliant( F, Lf),
                     ok( Lp, Lf),
                      (imprime(1),write([compatpigli,1,[F,S,Lf,Ls]]),
                       write('.'),nl;true).
compatpigli(F,S) :- extpig( S, Lp),
                    extliant( F, Lf),
                   (imprime(1),write([compatpigli,echec,[F,S,Lf,Ls]]),
```

```
                    write('.'),nl,false;false).

ok( L, Liant) :- saponifiable(A),
                 not(appartient( Liant, A)),
                 (imprime(1),write([ok,1,[L]]),
                 write('.'),nl;true).

ok([], L) :- (imprime(1),write([ok,2,[L]]),
              write('.'),nl;true).

ok( [T|Q], Liant) :- compzinc(A),
                     not(appartient(T, A)),
                     ok( Q, Liant),
                     (imprime(1),write([ok,3,[L]]),
                     write('.'),nl;true).

ok(L,_) :- (imprime(1),write([ok,echec,[L]]),
            write('.'),nl,false;false).


/* 1. CONSTRUCTION DE LA OU LES SOLUTIONS                          */

rechcom(E2,0).

rechcom(E2,X) :-
                     assert(sol(E2)),
                     tell(fichatt),
                     repeti(X),
                     told.

repeti(X) :-              repeat,
                            liresol(L),
                            detnoeuddev(L,N),
                            selectrtlarg(N),
                            liresol(L5),
                            liresolterm(L2),

                            (stop(X,L2,L5),!,true;fail).

 /* PASSAGE     */

passage(E1,E2,L) :- optimise(E1),
                    appel(L),
                    enleve.


appel(L) :-repeat,
                  solution(L1,E1,[Ef,[]]),
                  (L1 = [],lectsolu(L),!,true;
                   assert(solu(L1)),fail).


solution([L1|L2],E1,[Ef,[]]) :-
```

```
                    etat_de_depart(E1),
                    etat_final(Ef),
                    resolution([L1|L2],E1,[Ef,[]]).

solution([],E1,E2).

resolution([],E1,E2) :- term(E1,E2),!.

resolution([L1|L2],E1,[Ef,[]]) :-
                post(L1,Es),
                convient(L1,Es,Ef),

                pre( L1, Ep),

                extcarspec( Ep, Carp),
                extetatsurf( Ep, Esp),
                extas( Ep, Asp),
                extres( Ep, Resp),
                extprot( Ep, Prop),
                extderntrt( Ep, Dtp),

                extap( Es, Aps),
                extas( Es, Ass),
                extres( Es, Ress),
                extprot( Es, Pros),
                calc([Ef,[]],Ec , L1, Aps, Ass, Ress, Pros,
Carp,  Esp,  Asp,  Resp,  Prop, Dtp),
                    resolution(L2,E1,Ec).



/* 2. SELECTION D'UNE SOLUTION PARTIELLE                              */

detnoeuddev([],[]).
detnoeuddev([A|L],I) :- cher([A|L],A,I,2),
                        retract(sol(I)).

/* selection du traitement en largeur */
selectrtlarg(N) :- etat_de_depart(Ed),!,
                repeat,  cherchtrtok(Trt,Ed,N),
                         (Trt = [], true ;
                          cherchparam(Trt,  Carp, Esp, Asp, Resp,
Prop, Dtp, Aps, Ass, Ress, Pros),
                          calculn(N, Etatc, Trt, Aps, Ass, Ress, Pros,
Carp,  Esp,  Asp,  Resp,  Prop, Dtp),
                           memoire(Ed,Etatc),fail),!.


/*  3. CONTINUATION DE LA RECHERCHE                                   */

stop(_,_,[]) :- !.
stop(X,Lt,Ln):-
                nombre(Lt,Y),
```

```
                    Y > X,
                    rech(Lt,E,1),
                    extev(E,Ev),
                    rech(Ln,El,2),El = [_,E2],
                    extev(E2,Evn),
                    Ev < Evn ,!.


/*          4. SELECTION D'UN TRAITEMENT                                  */

cherchtrtok(Trt,Ed,[Ec,Ec1]) :-
                    effetdebord(Trt,[[pm,_],          [tsech,_],
[comp,_], [base,_], [prom,_], [fac,_], [stock,1], [odeur,_],
[anci,_], [maplir,_], [maplip,_], [qual,_]]),
                    post(Trt,Es),
                    aide(Trt,Es,[Ec,Ec1]).


cherchtrtok([],Ed,[_,Ec]) :- extsol(Ec,S),
                             enregistre(5,S).

aide(Trt,Es,[Ec,Ec1]) :- convient(Trt,Es,Ec),!.
aide(Trt,Es,[Ec,Ec1]) :-
                          extitemsvect( Es, Nats, Formes, Envs, Aps,
                                        Cars, Ess, Ass, Ress, Pros, Dts)
                 extitemsvect( Ec, Natf, Formef, Envf, Apf, Carf, Esf, As
                               Resf, Prof, Dtf),

                 faitavancer( Aps, Apf, Cars, Carf, Ass, Asf, Ress, Resf,
                              Pros, Prof),
                 extsol(Ec1,Sol),
                 concat(S,Sol,[Trt]),
                 enregistre(4,S),!,fail.

memoire(Ed,Etatc) :-
                    (term(Ed,Etatc),

                  Etatc =[_,[[sol,S], [pm,P], [comp,Co], [prom,Po],
[tsech,Ts], [qual,Q], [anci,A], [fac,F], [maplir,R], [maplip,Pi],
[odeur,O], [npro,Npr], [evalu,Ev]]],

                    assert(solterm([[sol,S], [qual,Q], [anci,A],
[maplir,R], [maplip,Pi], [odeur,O], [evalu,Ev]]));
                    assert(sol(Etatc))),!.


term([[nature, A], [forme, B], [env, C], [applipart, M],
[carspec, D],[etatsurf,  E],  [aspect,  F], [resist,  G],
[protect, H],[derntrt, Y]],
       [[[nature,  A], [forme, B], [env, C], [applipart,
[]],[carspec,  D],[etatsurf,  I], [aspect,  X], [resist,  G],
[protect, H],[derntrt, Z]],_]) :-

                 convientes(I, E),
                 extlistitem(perm, aspect, X, Perm),
```

```
                diff(X, Perm, F),!.

cherchparam(Trt,  Carp, Esp, Asp, Resp,Prop, Dtp, Aps, Ass, Ress,
Pros) :-
                post(Trt, Es),

                pre( Trt, Ep),

                extcarspec( Ep, Carp),
                extetatsurf( Ep, Esp),
                extas( Ep, Asp),
                extres( Ep, Resp),
                extprot( Ep, Prop),
                extderntrt( Ep, Dtp),

                extap( Es, Aps),
                extas( Es, Ass),
                extres( Es, Ress),
                extprot( Es, Pros).


/*      5. PASSAGE AU NOUVEAU VECTEUR GENERAL                          */

calculn([N,N1],[ Ec, Ec1],  Trt,  Aps, Ass, Rs, Pros,Ca, Esp, Asp, Rp,
Prop, Dt) :-

                Ec = [[nature,  Na], [forme,  For], [env,
Envf],  [applipart,   Apc],[carspec,   Ca], [etatsurf,  Esp],
[aspect, Asc],[resist, Resc], [protect,Proc], [derntrt, Dt]],
                extas( N, Asf),
                extres( N, Rf),
                extprot( N, Prof),
                extap( N, Apf),
                etat_final( Ef),
                extnat( Ef, Na),
                extforme(Ef, For),
                extenv( Ef, Envf),
                extap( Ef, Apf),

                  calculap(Apc, Aps, Apf),!,

                  calculas(Asc, Ass, Asp, Asf), !,


    extlistitem(perm, resist, Rf, Pr),
    calculres(Resc, Rf, Rs, Rp, Pr),!,
    extlistitem(perm, protect, Prof, Pp),
    calculaux(Pp, Prof, Pros, Prop, Proc),!,
    calculec(Ec1, Ec, Trt, N1), !.
calc([N,[]],[Ec,[]],  Trt,  Aps, Ass, Rs, Pros,Ca, Esp, Asp, Rp,
Prop, Dt) :-

                Ec = [[nature,  Na], [forme,  For], [env,
Envf],  [applipart,   Apc],[carspec,   Ca], [etatsurf,  Esp],
```

```
[aspect, Asc],[resist, Resc], [protect,Proc], [derntrt, Dt]],
                extas( N, Asf),
                extres( N, Rf),
                extprot( N, Prof),
                extap( N, Apf),
                etat_final( Ef),
                extnat( Ef, Na),
                extforme(Ef, For),
                extenv( Ef, Envf),
                extap( Ef, Apf),

                  calculap(Apc, Aps, Apf),

                  calculas(Asc, Ass, Asp, Asf),


                extlistitem(perm, resist, Rf, Pr),
                calculres(Resc, Rf, Rs, Rp, Pr),
                extlistitem(perm, protect, Prof, Pp),
                calculaux(Pp, Prof, Pros, Prop, Proc),!.


calculec(Ec1, Ec ,T, N1) :-
                Ec1 = [[sol,S],  [pm,P],[comp,Co], [prom,Po],
[tsech,Ts],      [qual,Q],[anci,A]      ,      [fac,F],[maplir,R],
[maplip,Pi],[odeur,O], [npro,Npr], [evalu,Ev]],

                  effetdebord(T,Eb),
                extprom(Eb,Pob),
                extfac(Eb,Fb),
                extnpro(N1,Np),
                extprom(N1,Poc),
                extfac(N1,Fc),
                extsol(N1,Sol),

                  concat(S,Sol,[T]),
                  Npr is Np + 1,
                  calcodancimeth(O,A,Pi,R,Eb,N1),
                  caltsprixcom(Ec ,Ts ,P ,Co ,Q , Npr,Eb, N1),
                  Po is Poc + Pob,
                  (Fc = oui, F = Fb;
                              F = Fc),!,
                  calculev(Ec,Ev,[P,Co,Po,Ts,F,Npr]).


calcodancimeth(O, A, Pi, R,Eb,N1) :-
                extodeur(Eb,Ob),
                extanc(Eb,Anb),
                extpist(Eb,Pb),
                extroul(Eb,Rb),
                extodeur(N1,Oc),
                extanc(N1,Anc),
                extpist(N1,Pc),
                extroul(N1,Rc),
                ( Oc = o, O = o;
```

```
                    O = Ob),
                ( Anc = n, A = n;
                           A = Anb),
                ( Pc = [], Pi = n;
                           Pi = Pb),
                ( Rc = [] , R = n;
                            R = Rb).


caltsprixcom(Ec, Ts, P, Co, Q, Npr,Eb,N1) :-

                extprix(Eb,Pb),
                exttsech(Eb,[Ts1,Ts2]),
                extcomp(Eb,Cob),
                extprix(N1,Pc),
                exttsech(N1,Tsc),
                extcomp(N1,Coc),

                  etat_de_depart(Ed),
                  (term(Ed,[Ec,_]),
                    (Npr = 1, P is (Pb * 3),
                              Ts is (( 2 * Ts2) + Ts1);
                    (Npr = 2, P is ((2 * Pb) + Pc),
                              Ts is (( 2 * Ts2) + Tsc);
                     P is Pb + Pc,
                     Ts is (Ts2 + Tsc)));
                  (Npr = 1, P is Pb,
                            Ts = Ts1;
                   P is Pb + Pc,
                   Ts is Ts2 + Tsc)),
                   (Cob = bi, Co = 1;
                              Co = Coc),
                    (Npr = 1, extqual(Eb,Q);extqual(N1,Q)).


calculas(Asc, Ass, Asp, Asf) :- extlistitem(supp, aspect, Asf, Supp),
                extlistitem(perm, aspect, Asf, Perm),
                calculaux(Perm, Supp, Ass, Asp, Asc).

calculaux(Perm, Supp, Moins, Pre, Res) :-
                  diff(Supp, Moins, Diffp),
                  concat(Tem, Diffp, Perm),
                  concat(Resul, Tem, Pre),
                  dedouble(Res, Resul).

calculap(Apc, Aps, Apf) :- diff(Apf, Aps, []),
                           concat(Apc, [], []);
                           concat(Apc, Apf, []).

calculres(Resc, F, S, P, Perm) :- calculresaux( N, F, S),
                                  concat(Tem, N, Perm),
                                  concat(Resu, Tem, P),
                                  dedouble(Resc, Resu).

calculresaux([],[], S).
```

```
calculresaux(N, [Rf|F], Rs) :- croiss(resist, L),
                               appartient(Rf, L),
                               appartient(R, Rs),
                               appartient(R, L),
                               calculresaux(N, F, Rs), !.
calculresaux([R|N], [R|F], Rs) :-calculresaux(N, F, Rs), !.

/*                    EVALUATION DE LA FONCTION                    */

calculev(Ec,Ev,Donnee) :-
        etat_de_depart(Ed),
        not(term(Ed,[Ec,_])),
                Ec = [[nature, Na], [forme,_], [env,
Envf], [applipart, Apc],[carspec, Ca], [etatsurf,_],
[aspect, Asc],[resist, Resc], [protect,Proc], [derntrt,_]],
              eval(Na,Envf,Apc,Ca,Asc,Resc,Proc,Maxi),
              somme(Donnee,Maxi,Resu),
              evalue(Ev,Resu).


calculev(_,E,[P,Co,Pr,Ts,F,N]) :-

      coeffimp([[pm,Pi],[comp,C], [prom,Pro], [tsech,Te],
[fac,Fa], [odeur,_], [anc,_], [maplir,_], [maplip,_] ,[qual,_],
[npr,Ni]]),
      coeffpond([[pm,Po],[comp,Cpo], [prom,Ppo], [tsech,To],
[fac,Fpo], [odeur,_], [anc,_], [maplir,_], [maplip,_] ,[qual,_],
[npr,No]]),
       E is ((P * Pi * Po) +(Co * C * Cpo) +(Pr * Pro * Ppo)
+(Ts * Te * To) +(N *Ni * No)).


eval(Na,Envf,Apc,Ca,Asc,Resc,Proc,[Pm,Com, Pom, Tsm, Fm]) :-

          evala(Na,Envf,Apc, Ca, Pa, Coa, Poa, Tsa, Fa),
          evalb(Asc, Resc, Proc,Pb, Cob, Pob, Tsb, Fb),
          max([Pa,Pb],Pm),!,
          max([Tsa,Tsb],Tsm),!,
          maxcomp([Coa,Cob],Com),!,
          max([Poa,Pob],Pom),!,
          maxfac([Fa,Fb],Fm),!.

evala(N,E,A,Coa,Pa,Ca,Pr,Ta,Fa) :-

          maxcl(N,[ [pm,P], [tsech,T], [prom,M], [comp,C],
[fac,F]]),
          maxcl(E,[ [pm,Pe], [tsech,Te], [prom,Me], [comp,Ce],
[fac,Fe]]),
          maxcl(A,[ [pm,Ps], [tsech,Ts], [prom,Ms], [comp,Cs],
[fac,Fs]]),
          maxcl(Coa,[ [pm,Pc], [tsech,Tc], [prom,Mc],
[comp,Cc], [fac,Fc]]),
          max([P,Pe,Ps,Pc],Pa),!,
          max([T,Te,Ts,Tc],Ta),!,
```

```
            maxcomp([C,Ce,Cs,Cc],Ca),!,
            max([M,Me,Ms,Mc],Pr),!,
            maxfac([F,Fe,Fs,Fc],Fa),!.

evalb(A, R, P,Pb, Cob, Pob, Tsb, Fb) :-

            maxcl(A,[    [pm,Pa],       [tsech,Ta],    [prom,Poa],
[comp,Ca], [fac,Fa]]),
            maxcl(R,[    [pm,Pr],       [tsech,Tr],    [prom,Por],
[comp,Cr], [fac,Fr]]),
            maxcl(P,[    [pm,Pp],       [tsech,Tp],    [prom,Pop],
[comp,Cp], [fac,Fp]]),
            max([Pa,Pr,Pp],Pb),
            max([Ta,Tr,Tp],Tsb),
            maxcomp([Ca,Cr,Cp],Cob),
            max([Poa,Por,Pop],Pob),
            maxfac([Fa,Fr,Fp],Fb).

maxcl([],[   [pm,0],    [tsech,0],   [prom,0],   [comp,mono],
[fac,oui]]).

maxcl([L1|L2],[   [pm,P],       [tsech,T],    [prom,Po],   [comp,C],
[fac,F]]):-
           maxcl(L2,[    [pm,P2],       [tsech,T2],    [prom,Po2],
[comp,C2], [fac,F2]]),
           classe(L1,[    [pm,P1],       [tsech,T1],    [prom,Po1],
[comp,C1], [fac,F1]]),
            max([P1,P2],P),!,
            max([T1,T2],T),!,
            maxcomp([C1,C2],C),!,
            max([Po1,Po2],Po),!,
            maxfac([F1,F2],F),!.
maxcomp([],mono).
maxcomp([L1|L2],Max) :- maxcomp(L2,M),
                        (L1 = mono, Max = M;Max = L1).
maxfac([],oui).
maxfac([L1|L2],Max) :- maxfac(L2,M),
                       (L1 = oui, Max = M; Max = L1).

somme([Pr,  Co,  Pro,  Ts, Fac, Np], [Pm, Com, Pom, Tsm, Fm], [P1, C1,
Pr1, Ts1, Fa1, Np1]) :-
           P1 is Pr + Pm,
           (Com = mono, C1 = Co;
                        C1 = 1),
           Pr1 is Pro + Pom,
           Ts1 is Ts + Tsm,
           (Fm = oui, Fa1 = Fac;
                      Fa1 = 1),
           Np1 is Np + 1,
           (imprime(1),write([somme,1,[[Pr,Co,Pro,Ts,Fac,Np],[Pm, Com,
           Pom,    Tsm,    Fm],    [P1,    C1,    Pr1,    Ts1,    Fa1,
           Np1]]]),
           write('.'),nl;true).

evalue(Ev,[P1, C1, Pr1, Ts1, Fa1, Np1]) :-
```

```
      coeffimp([[pm,Pi],[comp,C], [prom,Pro], [tsech,Te], [fac,Fa],
[odeur,_], [anc,_], [maplir,_], [maplip,_] ,[qual,_], [npr,Ni]]),
      coeffpond([[pm,Po],[comp,Cpo], [prom,Ppo], [tsech,To],
[fac,Fpo], [odeur,_], [anc,_], [maplir,_], [maplip,_] ,[qual,_],
[npr,No]]),
       Ev is ((P1 * Pi * Po) +(C1 * C * Cpo) +(Pr1 * Pro * Ppo) +(Ts1
* Te * To) +(Np1 *Ni * No)).


/*   B.3.     MOTEUR D'INFERENCE                                    */

rech([],[],_).
rech([A|L],I,W) :- cher([A|L],A,I,W).

cher([],I,I,_).
cher([A|L],Max,I,W) :- sup(A,Max,W),
                       cher(L,Max,I,W).
cher([A|L],_,I,W) :- cher(L,A,I,W).


/*  7.  LE QUESTIONNAIRE                                           */

choix(L,X,L) :- nombre(L,Y),
                Y =< X,
                tell(cat2),ecrirefich([]),told.

choix(L,X,R) :- tri(L,Ltri,1),
                cherchmaxmin(X,Ltri,Max,Min),
                choixquest(Ltri,Quest),
                recherchso(Quest,Som),
                Emax is Max - Som, Emin is Min + Som,
                prisesyst(L,Ens,Emax,Emin),
                choixquest(Ens,Questi),
                tri(Questi,Questrie,4),
                calcquest(Ltri,Questrie,Valens,Max,Min,X),
                prise(Valens,X,R).


prise(_,0,[]).
prise(L,X,R) :- prend(L,X,R,0).

prend(L,X,[],X) :- tell(cat2), extraire(L,Lbis),
                   ecrirefich(Lbis),told,!.
prend([L1|L],X,[L1|R],Y) :-
          Y1 is Y + 1,
          prend(L,X,R,Y1).

enregistre(Cat,E) :- write([Cat,E]),write('.'),nl.

calcquest(Ltri,Quest,Ltri,Max,Min,X) :-
                utilitequesti(Min,Max,Quest,stop).

calcquest(L,[Qu|Quest],Inc,Max,Min,X) :-
```

```
                questionnaire(Qu,Val,Reponse),
                donneval(L,Reponse,Qu,Val,Lval),
                tri(Lval,Ltri,1),
                cherchmaxmin(X,Ltri,Emax,Emin),
                calcquest(Ltri,Quest,Inc,Emax,Emin,X).


recherchso([],0).
recherchso([Q1|Q2],Som) :-
        recherchso(Q2,S1),
        extev(Q1,Val),
        Som = S1 + Val.

/*   8. LE CHOIX DES QUESTIONS                                    */


choixquest(Ltri,Quest) :-
        lirequestion(Q),
        questchoix(Ltri,Questi,Q),
        tri(Questi,Quest,4).

questchoix(L,[],[]).
questchoix(L,[Q1|Q2],[Q1|Squ]) :-
        extvalcrit(car,Q1,Car),
        utile(L,Car),
        questchoix(L,Q2,Squ).

questchoix(L,Q2,[Q1|Squ]) :-
        questchoix(L,Q2,Squ).


utile([L1|L],Car) :- extvalcrit(Car,L1,Vali),
                     not(egalut(L,Vali,Car)).

egalut([],Vali,Car).
egalut([L1|L],Vali,Car) :- extvalcrit(Car,L1,Vali),
                           egalut(L,Vali,Car).


/* 9.  POSER LA QUESTION A L'UTILISATEUR                         */


questionnaire(Qu,Val,Reponse) :-
        extvalcrit(quest,Qu,Ph),
        extvalcrit(val,Qu,Val),
        poser(Ph,Reponse).

/* 10.   CALCUL DE L'EVALUATION DES SYSTEMES                     */


donneval([],_,_,_,[]).
donneval([L1|L],o,Q,Val,[[[sol,S], [qual,Qual],[anci,A], [maplir,R],
[maplip,Pi],[odeur,O], [evalu,Evalu]]|Lvalu]):-
        L1 = [[sol,S],      [qual,Qual],[anci,A],[maplir,R],[maplip,Pi]
[odeur,O], [evalu,Ev]],
```

```
        extev(Q,Eval),
        extvalcrit(car,Q,Car),
        extvalcrit(Car,L1,Val),
        Evalu is Ev + Eval,
        donneval(L,o,Q,Val,Lvalu).

donneval([L1|L],Rep,Q,Val,[L1|Lvalu]) :-
        donneval(L,Rep,Q,Val,Lvalu).


/*  10.  CONTINUATION DU QUESTIONNAIRE                                  */


utilitequesti(Min,Max,Quest,stop) :-
        somqu(Quest,Res),
        Verif is Max + Res,
        (Verif < Min ; Verif = Min).

utilitequesti(_,_,_,continuer).

somqu([],0).
somqu([Q1|Q],Res) :-
        somqu(Q,R1),
        extev(Q1,Ev),
        Res is R1 + Ev.

cherchmaxmin(X,Ltri,Max,Min) :- lecturex(X,Ltri,Max,Min,1).

lecturex(X,[Emax|E2],Max,Min,X) :-
                E2 = [Emin|_],
                extev(Emax,Max),
                extev(Emin,Min).

lecturex(X,[E1|E2],Emax,Emin,Y)  :-
                Y1 is Y + 1,
                lecturex(X,E2,Emax,Emin,Y1).

/*   11. SELECTION DES SYSTEMES A DEPARTAGER                            */

prisesyst([],[],_,_).
prisesyst([L1|L2],[L1|L3],Max,Min) :-
        extev(L1,Ev),
        Ev < Min, Ev > Max,
        prisesyst(L2,L3,Max,Min).

prisesyst([L1|L2],L3,Max,Min) :- prisesyst(L2,L3,Max,Min).



travail(1) :- entete,entdonnees,
              sortphr(['Nombre',de,solution,'desiree.']),
              read(X),
              retract(etat_de_depart1(Ed)),assert(etat_de_depart(Ed)),
              retract(etat_final1(Ef)),assert(etat_final(Ef)),
              solution(Pre,Ed,[Ef,[]]),sortieprepa(Pre),
```

```
                retract(etat_final(Ef)),retract(etat_de_depart(Ed)),
                retract(etat_de_depart2(Ede)),assert(etat_de_depart(Ede)),
                retract(etat_final2(A)),assert(etat_final(A)),
                etat_final_com(Q),
                rechcom([A,Q],X),
                liresolterm(Lterm),
                choix(Lterm,X,Solaff),!,extraire(Solaff,Soluaff),
                sortie(Soluaff),
                see(fichatt),
                lire(Fic),
                seen,
                verif(Fic,L4,L5),
                tell(cat4),
                ecrirefich(L4),
                told,
                tell(cat5),
                ecrirefich(L5),
                told,
                tell(cat3),
                liresol(L3),extraction(L3,L3bis),
                ecrirefich(L3bis),
                told,
                tell(cat1),ecrirefich(Soluaff),told.

extraire([],[]).
extraire([A|B],[Sol|C]) :- extsol(A,Sol),
                           extraire(B,C).

verif([],[],[]).
verif([[5,L1]|L],L4,[L1|L5]) :- verif(L,L4,L5).
verif([[4,L1]|L],[L1|L4],L5) :- verif(L,L4,L5).
extraction([],[]).
extraction([[Ec,Ecc]|L],[Sol|C]) :- extsol(Ecc,Sol),
                                    extraction(L,C).

/*   1. DESIGNATION DE LA NATURE                                          */

travail(2) :- read(S),
              visionhisto(S,Cat,Sysfich,Sysrendu),
              elecompl(S,Sysfich,Sysrendu,Etat),
              explniv(Cat,Etat,S,Sysfich,Sysrendu),
              (retract(pointeur(A,B)),assert(pointeur(2,Sysfich));
               assert(pointeur(2,Sysfich))).


/* 1.1.  LA RECHERCHE HISTORIQUE                                         */


visionhisto(S,Cat,B,Sysrendu) :-
                        ( lectfich(cat1,L),
                          egalsys(S,L,B),
                          Cat = 1,Sysrendu = S;
                          lectfich(cat2,L),
                          egalsys(S,L,B),
                          Cat = 2,Sysrendu = S;
```

```
                              lectfich(cat3,L),
                              egalsys(S,L,B),
                              Cat = 3,Sysrendu = S;
                              lectfich(cat4,L),
                              egalsys(S,L,B),
                              Cat = 4,Sysrendu = S;
                              lectfich(cat5,L),
                              egalsys(S,L,B),
                              Cat = 5,Sysrendu = S).

visionhisto(S,Cat,A,Sysrendu) :- enlevelele(S,S1,_),
                                 (S1  =[],Sysrendu = [],A = [],Cat = 5;
                                 visionhisto(S1,Cat,A,Sysrendu)).


egalsys(A,[B|X],B) :- verifa(A,B).
egalsys(A,[X|L],C) :- egalsys(A,L,C).

verifa([],X).
verifa([A|Sui],[A|B]) :- verifa(Sui,B).


/*  1.2.  L'IDENTIFICATION DE LA NATURE                                  */


elecompl(S,Sysfich,Sysrendu,Etat) :- nombre(S,X),
                        nombre(Sysfich,Y),nombre(Sysrendu,Z),
                        etat(X,Z,Y,Etat).

etat(X,X,X,1).
etat(X,X,_,2).
etat(_,Y,Y,3).
etat(_,_,_,4).

/*  2. LA TRACE LOCALE                                                   */

/*  2.1.  CONSTRUCTION DE LA TRACE LOCALE                                */

traceloc(Syst) :- (imprime(1);assert(imprime(1))),
                  etat_final(N1),
                  etat_final_com(N2),
                  tell(trace),
                  tracetrt(Syst,[N1,N2]),
                  told.

tracetrt([],Etatc) :- write(Etatc),write('.'),nl.

tracetrt([Trt|L],[N1,N2]) :-
            write(Trt),write('.'),nl,
            write([N1,N2]),write('.'),nl,
            post(Trt,Es),
            (convient(Trt,Es,N1),
            cherchparam(Trt,  Carp,  Esp, Asp, Resp, Prop,Dtp,Aps,
Ass, Ress, Pros),
            calc([N1,[]],Etatc, Trt, Aps, Ass, Ress, Pros, Carp, Esp,
```

```
Asp, Resp, Prop, Dtp);true),
          write('chtrt'),write('.'),nl,
          tracetrt(L,Etatc).

/*  2.2.  OUTILS POUR L'EXPLOITATION DE LA TRACE LOCALE                  */


unitetrt(Trt,[],[],[]).
unitetrt(Trt,[Trt|Sui],Trttrace,Etat) :- [Etat|Suite] = Sui,
                                         remplir(Suite,Trttrace).
unitetrt(Trt,[A|Sui],Trttrace,Etat) :- suite(Sui,Suite),
                                       unitetrt(Trt,Suite,Trttrace,Etat)

remplir(['chtrt'|Sui],[]).
remplir([A|Sui],[A|B]) :- remplir(Sui,B).

suite(['chtrt'|Sui],Sui).
suite([_|Sui],Suite) :- suite(Sui,Suite).


recherche(Re,[],[]).
recherche(Re,[[Re,A,B]|_],[Re,A,B]).
recherche(Re,[A|Sui],Ele) :- recherche(Re,Sui,Ele).

etude(Re,Trttrace,[[A,B,C]|List]) :-
            recherche(Re,Trttrace,[A,B,C]),
            enfant(A,Enfant),
            recherchenfant(Enfant,Trttrace,List,B).


recherchenfant([],_,[],R).
recherchenfant([E1|E],Trttrace,[X|L],R) :-
          recherche(E1,Trttrace,X),
          (([A,B,C] = X,R = echec,B = echec ;
            [A,B,C] = X,not(R = echec),not(B = echec))
            ;true),
          recherchenfant(E,Trttrace,L,R).

etupar(Regle,Trttrace,Liste) :- parent(Regle,Parent),
                                recherche(Parent,Trttrace,Liste).

etupar(_,_,[]).

/* 3. L'EXPLICATION COMMERCIALE                                          */

travail(8) :- pointeur(_,Syst),
              visionhisto(Syst,Cat,_,_),
              ( not (Cat = 4), not(Cat = 5),
               explicom(Syst),
               menucom(Rep),
               (Rep = 1,
                (retract(pointeur(2,Syst)),
                 traceloc(Syst),
                 assert(pointeur(3,Syst));true),
                expl2com(Cat,Syst);true);
```

```
                sortphr(['le',systeme,Syst,'n''est',pas,correct,'techniqu
                ).


/* 4.  L'EXPLICATION GENERALE DU SYSTEME                                  */

travail(4) :- ((pointeur(2,Syst),
                traceloc(Syst);
                 pointeur(3,Syst)),
                lectfich(trace,Trace),
                expli2niv(Syst,Trace),
                retract(pointeur(_,Syst)),
                assert(pointeur(3,Syst));
                sortphr(['Vous',ne,pouvez,pas,posez,cette,'question.'])).


expli2niv([],Trace) :- sortphr(['Et',nous,arrivons,a,'l''etat:']),nl,
                       enleveiele(Trace,_,Etat),
                        presentationetat(Etat).

expli2niv([Trt|Syst],Trace) :-
           unitetrt(Trt,Trace,Trttrace,Etat),
           sortphr(['L''etat',du,support,'est:']),
           presentationetat(Etat),
           sortphr(['Le',traitement,Trt]),
           etude(faitavancer,Trttrace,Liste),
           sortiefam(Liste),
           [L1|_] = Liste,
           (L1 = [_,echec,_];
            recherche(convient,Trttrace,Ele),
            francais(Ele,Phrase),
            sortieiniv(Phrase)),
            expli2niv(Syst,Trace).

/*  5.  PLUS DE PRECISIONS SUR CERTAINES REGLES                          */

travail(5) :- (pointeur(3,Syst),
                questionregletrt(X,Trt),
                traduction(X,Regle),
                lectfich(trace,Trace),
                unitetrt(Trt,Trace,Trttrace,_),
                recherche(Regle,Trttrace,[A,B,C]),
                francais([A,B,C],Phrase),
                sortieiniv(Phrase)
                (enfant(A,Enf),
                 recherchenfant(Enf,Trttrace,Lis,B),
                 sortphr([en,fonction]),nl,
                 suit(Lis);true);
                sortphr(['Vous',ne,pouvez,pas,posez,cette,'question.'])),
                attendre.


travail(6) :- (pointeur(3,Syst),
                questionregletrt(X,Trt),
                traduction(X,Regle),
```

```
                lectfich(trace,Trace),
                unitetrt(Trt,Trace,Trttrace,_),
                etupar(Regle,Trttrace,[A,B,C]),
                sortiepar([A,B,C])
                (enfant(A,Enf),
                 recherchenfant(Enf,Trttrace,Lis,B),
                 sortphr([en,fonction],
                 nl,suit(Lis);true);
                sortphr(['Vous',ne,pouvez,pas,posez,cette,'question.'])),
                attendre.

verifprod(P,[P|L]).
verifprod(P,[_|L]) :- verifprod(P,L).

  /*   6. EXPLICATION DE L'UTILITE D'UN TRAITEMENT                  */

travail(7) :- read(P),
              (lectfich(cat1,L),
               egalprod(L,P,Sys),
               sortphr(['Le',produit,P,est,present,dans,le,systeme,Sys
                        ,'.']);
               lectfich(cat2,L),
               egalprod(L,P,Sys),
               sortphr(['Le',produit,P,est,present,dans,le,systeme,Sys
                        ,'.']),
               sortphr(['Ce',systeme,'n''a',pas,ete,repris,pour,des,rais
               lectfich(cat3,L),
               egalprod(L,P,Sys),
               sortphr(['Le',produit,P,est,present,dans,le,systeme,Sys
                        ,'.']),
               sortphr(['Ce',systeme,'n''est',pas,un,systeme,'final.']);
               lectfich(cat4,L),
               egalprod(L,P,Sys),
               sortphr(['Le',produit,P,est,present,dans,le,systeme,Sys
                        ,'.']),
                sortphr(['Ce',systeme,ne,convient,pas,'techniquement.'])
               sortphr(['Votre',produit,P,'n''est',pas,'valable.'])),
               nl.


/*    7. EXPLICATION DE L4INEXISTANCE DE SOLUTION                    */

travail(3) :- (lectfich(cat1,A),A = [],
               lectfich(cat3,A3),
               lectfich(cat4,A4),
               lectfich(cat5,A5),
               concat(Li,A3,A4),
               concat(Liste,Li,A5),
               sortphr(['On',ne,peut,pas,aller,plus,loin,'que:']),
               nl,
               sort(Liste);
               sortphr(['Il',existe,au,moins,une,'solution.']),!).
```

```
travail(9).


/* ************************************************************************/
/*  PROGRAMME    PRINCIPAL                                                 */
/************************************************************************/
pro :- repeat,
              menu,
              lireA(A),
              trav(A),
              (A = 9,true;fail).

trav(A) :- travail(A),!.
lireA(A) :- read(A).
```

Données des règles pour l'explication

REGLES AYANT REUSSI	PARAMETRES	DONNEES GLOBALES	DONNEES LOCALES
CONVIENTES	S,F	Trt	
CONVIENTES	S	Trt	
CONVIENTAS			
CONVIENTAS	S,F	Trt	Perm, Remp
CONVIENTAS	F		
CONVIENTDT	S	Trt	
CONVIENTDT			
CONVIENTDT	S,F		
CONVIENTDT			
COMPATAIR	S		Pvc,Env
COMPATLIANT	F,S		Lf, Ls
COMPATSOLV	F,S		Lf, Ls
COMPATPIGLI	F,S		Lp, Lf
COMPATPVC	F,S		Pf, Ps
OK	L		
OK			
OK	T,Q		
AVANCAP	Aps	Trt	
AVANCC	Cars,F,Carf	Trt	
AVANCA	Ass, Asf	Trt	As, A
AVANCR	Ress, Resf	Trt	A
AVANCP	Pros, Prof	Trt	A
CONVIENTNAT			
CONVIENTNAT	F		
CONVIENTNAT	Natf, S	Trt	
CONVIENTFORME			
CONVIENTFORME	F		
CONVIENTFORME	S,Forme, F	Trt	L
CONVIENTENV			
CONVIENTENV	E		
CONVIENTENV	S, E, F	Trt	L

REGLES AYANT REUSSI	PARAMETRES	DONNEES GLOBALES	DONNEES LOCALES
CONVIENTAP	F	Trt	Perm
CONVIENTAP		Trt	Aps, Apf
CONVIENTAP			
CONVIENTCARSPEC	S , F	Trt	
CONVIENTRES			
CONVIENTRES	S, R, F	Trt	L

Système conseil pour la mise en peinture des bois

1°) Quelles sont les différentes classes de bois qui entrainent des traitements distincts (agglomérés, ...) ? Quels sont les différents types de bois appartenant à ces classes ?

2°) La forme, par exemple
- verticale ou oblique ou horizontale
- tubulaire ou angulaire ou plan
- surface vers le bas ou surface vers le haut

a-t-elle une importance pour le choix d'un traitement ?
Y a-t-il une dépendance entre la forme et les classes ?
Quelles sont les différentes formes (admissibles pour chaque classe) ?

3°) Quels sont les différents types d'environnements (en fonction de chaque classe) ?

4°) Quelles sont les différentes dégradations qu'un bois peut avoir ?
Peut-on les regrouper en familles selon les traitements qu'elles nécessitent ?
Qu'entendons-nous par non-dégradation en ce qui concerne le bois ?
Est-ce le contraire de dégradation ou cette notion comprend-elle d'autres caractéristiques ?

5°) Quels sont les différents aspects que le bois peut prendre après application d'un traitement ? (opaque, coloré, brillant, translucide, lisse, sans noeud, ...)

6°) Quelles résistances admet-on pour le bois ? (résistance à l'environnement, à l'humidité, ...)

7°) Existe-t-il d'autres caractéristiques pour le bois qui ne rentrent pas dans les questions précédentes ?

8°) Quels sont les préparations et les traitements pour le bois, leur raison d'être, leurs conditions d'applications et les caractéristiques qui en résultent pour le bois ? Pour chaque traitement (produit), quel est le nombre de couches à appliquer (cela dépend-il des exigences du client ou des exigences chimiques) et le temps de séchage ?

9°) Quelles sont les contraintes chimiques à respecter afin d'éviter que deux traitements ne se neutralisent ou ne produisent des réactions non désirées ?

10°) Comment déterminer la qualité d'un produit